



Cybersecurity Risks in Complex IoT Environments: Threats to Smart Homes, Buildings and Other Structures

Stephen Hilt, Numaan Huq, Martin Rösler, and Akira Urano

TREND MICRO LEGAL DISCLAIMER

The information provided herein is for general information and educational purposes only. It is not intended and should not be construed to constitute legal advice. The information contained herein may not be applicable to all situations and may not reflect the most current situation. Nothing contained herein should be relied on or acted upon without the benefit of legal advice based on the particular facts and circumstances presented and nothing herein should be construed otherwise. Trend Micro reserves the right to modify the contents of this document at any time without prior notice.

Translations of any material into other languages are intended solely as a convenience. Translation accuracy is not guaranteed nor implied. If any questions arise related to the accuracy of a translation, please refer to the original language official version of the document. Any discrepancies or differences created in the translation are not binding and have no legal effect for compliance or enforcement purposes.

Although Trend Micro uses reasonable efforts to include accurate and up-to-date information herein, Trend Micro makes no warranties or representations of any kind as to its accuracy, currency, or completeness. You agree that access to and use of and reliance on this document and the content thereof is at your own risk. Trend Micro disclaims all warranties of any kind, express or implied. Neither Trend Micro nor any party involved in creating, producing, or delivering this document shall be liable for any consequence, loss, or damage, including direct, indirect, special, consequential, loss of business profits, or special damages, whatsoever arising out of access to, use of, or inability to use, or in connection with the use of this document, or any errors or omissions in the content thereof. Use of this information constitutes acceptance for use in an "as is" condition.

Published by to:

Trend Micro Research

Written by:

**Stephen Hilt, Numaan Huq,
Martin Rösler, and Akira Urano**

Stock images used under license from
Shutterstock.com

For Raimund Genes (1963-2017)

Contents

4

1. Introduction

7

2. Complex IoT Environments

17

3. Attacks Against Complex IoT Environments

26

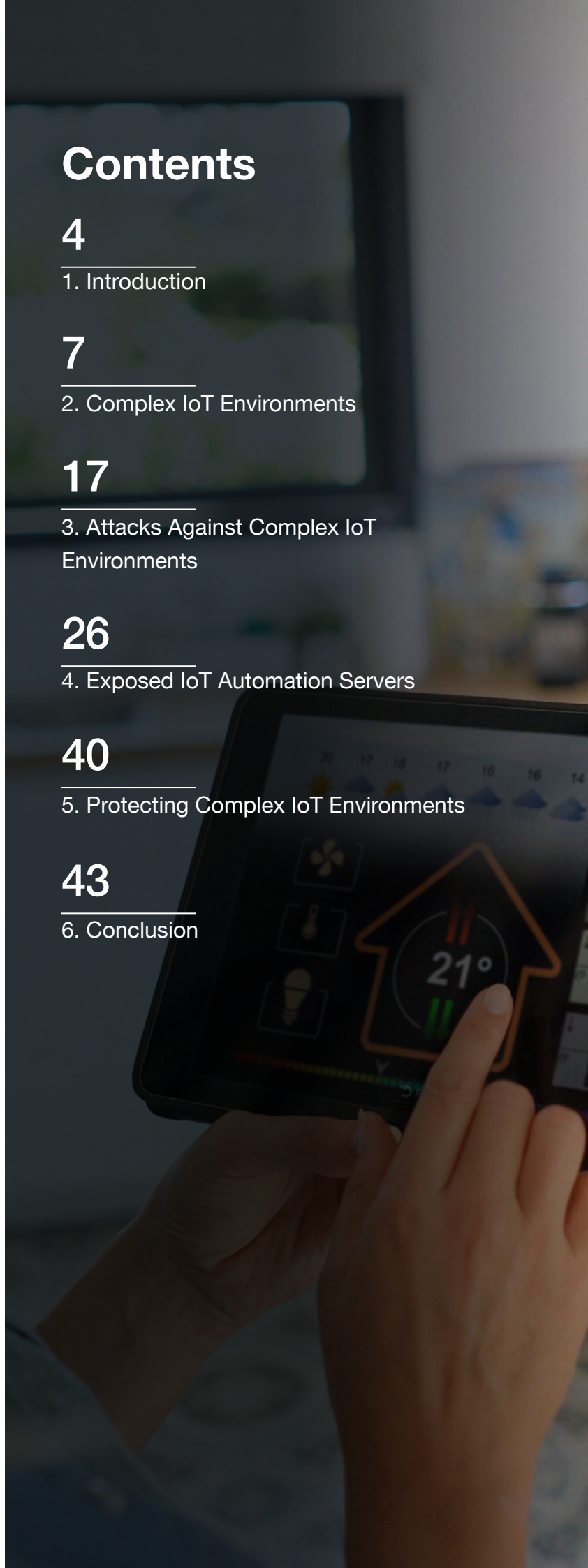
4. Exposed IoT Automation Servers

40

5. Protecting Complex IoT Environments

43

6. Conclusion



A person's hand is holding a smart thermostat in the foreground on the left. The thermostat screen shows a temperature of 22°C, a humidity of 65%, and a forecast of 17°C with a cloud icon. In the background, there is a window with a view of greenery outside and a potted plant on a wooden surface to the right.

Structures that integrate IoT technology are now evolving into what we call complex IoT environments (CIEs). We define CIEs as having a minimum of 10 internet of things (IoT) devices integrated into an environment using an IoT automation platform that functionally chains the devices together to create smart applications. The more familiar implementation of a CIE is the smart home, but CIEs are not restricted to that; IoT automation platforms can be scaled up to control devices in bigger structures like smart buildings.

Due to the increasing availability of IoT devices and accessibility of automation platforms, the adoption of smart homes — and smart buildings — is accelerating. The convenience and speed, however, come with risks. To enable IoT devices to simplify and improve our lives, rules written in automation platforms are becoming more and more complex. This opens up new attack avenues that haven't been explored in depth before because the main focus of smart home security research today has been on how individual IoT devices might be attacked.

We sought to fill this gap by setting up two smart home laboratories, one in Germany and one in the U.S., fitted with a wide variety of IoT devices and using different automation platforms. Through these smart home labs, we explored the working components of CIEs, that is, IoT automation servers and smart applications, as well as attack surfaces and common security problems.

In this paper, we discuss smart home setups, common automation platforms and supported protocols, and our smart home labs' configurations. We then describe potential risks and smart attacks against CIEs, including attacks on smart locks and speakers, exposed sensitive information, and injection of logic bugs. We also give a rundown of exposed and unsecured IoT automation servers that we found and possible security ramifications of such exposure online. Finally, we provide a set of guidelines on how to protect CIEs to help smart home owners, smart building administrators, and adopters in other settings and industries.

1. Introduction

“Ok Janet, please check if the house is locked and turn on the alarm after I leave,” a homeowner commands to a virtual home assistant that will do exactly as ordered. Sounds futuristic? No. Home automation to achieve this functionality is possible using off-the-shelf internet of things (IoT) devices available on the market today. The only caveat: There is a lot of do-it-yourself (DIY) technology-bootstrapping necessary to correctly configure and automate smart homes. At the heart of these DIY projects is an emerging class of software products called *IoT automation platforms*, which act as the brain for the smart home.

Smart homes are now evolving into what we call *complex IoT environments* (CIEs). We define CIEs as a minimum of 10 IoT devices integrated into an environment using an IoT automation platform that functionally chains the devices together to create smart applications. This is because the level of complexity increases exponentially with each addition of a device. We determined 10 devices in a single environment to be more or less the point at which the integrated complexity has grown to a high enough level that users would switch to managing their devices using outside assistance, i.e., an AI-enabled assistant. In addition, most smart households now have at least 10 IoT devices, with the number expected to grow.¹ Another hallmark of CIEs is the conversion of the traditional “dumb” appliances, for example, toasters, washing machines, dishwashers, etc., into smart appliances using a network-connected smart plug. CIEs are not restricted to just smart homes; IoT automation platforms can be scaled up to control devices in smart buildings, with some products, like the Node-RED,² even supporting out-of-the-box cloud integration to enhance this ability.

IoT automation platforms enable the average user to easily configure and execute complex behavioral rules to manage all the connected devices in an environment, thus allowing functional chaining to create smart applications. One example of a smart application that we implemented in our smart home lab double pressing an EnOcean³ switch beside the main door of the house checks whether the contact sensors of all the doors and windows in the house are closed, then sends an SMS to the user’s mobile phone reporting whether everything is closed or a particular door/window is open.

The big issue with automation rules is that, as more and more devices are added to an action, the rule becomes increasingly complex and prone to logical errors. Likewise, as the number of rules in the system increases, it becomes more challenging to manage, track, and debug actions, especially if there are functional overlaps between rules. Furthermore, having many different types of devices in the CIE, each with their own security flaws, exponentially increases the potential attack surfaces — essentially, we are now creating brand-new security risks that didn't exist pre-smart buildings.

The availability of affordable IoT devices and easy-to-configure IoT automation platforms is going a long way in accelerating our smart home/building future. But this convenience and speed come with a cost. To better understand the cybersecurity risks in CIEs, it is important to first identify the two main smart home/building setups:

- **Bolt-on smart homes/buildings:** These are and will be the vast majority of smart homes and buildings. They are mostly older structures but can also include new constructions. The entire house is not wired for internet access, and the majority of IoT devices in the house are connected to the internet via Wi-Fi. Wi-Fi extenders or mesh Wi-Fi routers are required to ensure full internet coverage throughout the house. In addition to Wi-Fi, IoT devices also communicate using other wireless protocols such as ZigBee,⁴ Z-Wave,⁵ Bluetooth,⁶ etc. As the name suggests, all smart devices are “bolted on” to the home network, which itself is also bolted on. In bolted-on smart homes, network capacity determines the total number of supported devices. Compared to the other setup type (purpose built), a bolt-on model is easier to implement. As technology changes, upgrades can be done cheaply and quickly.
- **Purpose-built smart homes/buildings:** These homes or buildings are purposely built to be smart, which means they'll have Ethernet or fiber running in every room. Wi-Fi extenders or mesh Wi-Fi routers can be connected in each floor or in a group of rooms to improve internet connectivity, but aren't necessary in this type of smart home. Just like in bolt-on smart homes, IoT devices in this type of setup can communicate either over the wired network or wirelessly using Wi-Fi, ZigBee, Z-Wave, Bluetooth, etc. The house typically has one or more patch panels to manage all the installed ports and wireless communications. As the name suggests, many of the core networking and/or wireless technologies, for example, Gigabit Ethernet switches and ZigBee or Z-Wave® receivers, are already built into the house, and additional installation like in bolt-on smart homes is no longer required. The initial investment for a purpose-built smart home is higher than that for a bolt-on type, but the return on investment (ROI) is good especially in terms of the ease of implementation of new functionality. The only major caveat: If in the future the core technology running such environments dramatically changes, then upgrading equipment and wiring can be both expensive and disruptive.

Back in 2013 we had explored the risks related to networking protocols based on the IEEE 802.15.4 technical standard.⁷ As detailed in that paper, flaws related to these protocols could allow attackers to monitor user activity as well as tinker with devices. Many of our findings then are reflected in our current study, which takes a greater focus on complexity of IoT environments.



There has been a lot of research published about the cybersecurity risks of IoT devices, but there is little or no research published about smart applications (chaining together 10 or more IoT devices using an automation platform) or smart attacks (a cyberattack which chains multiple IoT devices to create an attack condition). To address this, in this paper we:

- Explore the working components of complex IoT environments (= IoT automation servers + smart applications).
- Share what we found in our search for insecure IoT automation platforms exposed on the internet and explore possible security ramifications of leaving platform servers exposed online.
- Theorize and demonstrate smart attacks against CIEs by exploring logic bugs, attack surfaces, and common security problems.
- Provide a set of guidelines on how to protect CIEs in both smart homes and smart buildings.

As IoT devices and services like Amazon Alexa™ service,⁸ Google Assistant™ virtual assistant⁹ Philips® Hue,¹⁰ Smart TVs, smart plugs, and so on become commonplace in homes and buildings, and as more and more people start integrating their IoT devices using IoT automation platforms like FHEM,¹¹ Home Assistant,¹² openHAB,¹³ and Domoticz,¹⁴ brand-new attack surfaces will emerge. Now is the prime opportunity to study the cybersecurity ramifications of CIEs before cascading cybersecurity risks catch us unaware. One immediate concern is the uptick of IoT botnets in recent years (most notable being the Mirai attacks) which is a reflection of how cybercriminals are shifting their interest to smart devices,¹⁵ and whose next logical evolution will be smart attacks.

2. Complex IoT Environments

The smart home transformation is moving forward full steam. The number of IoT devices in homes is increasing year over year for various reasons: manufacturers adding internet connectivity to their regular product lines; cloud connectivity and cloud services becoming commonplace; falling development costs because of cheap IoT original equipment manufacturer (OEM) components; mature IoT ecosystem that makes application development faster and cheaper; fast and affordable mobile internet; people's desire to streamline or automate daily tasks to improve their lifestyle.

According to Li et al., there are three generations of smart homes¹⁶:

- 1st generation — Wireless technology and proxy server home automation approach.
- 2nd generation — Artificial intelligence (AI) controls electrical devices.
- 3rd generation — Robot buddy that can interact with human beings.

Today's smart home cannot be definitively categorized as belonging to any of the three generations. When one looks at currently available technology, it becomes obvious that today's smart home is an amalgamation of all three generations. In addition, many of the devices have generation 1, 2, and 3 capabilities, further blurring the lines.

To better articulate this ecosystem, we coined the term complex IoT environment (CIE). We define this term based on Steven Johnson's definition of complexity in his book "Emergence: The Connected Lives of Ants, Brains, Cities, and Software."¹⁷ A system is complex if its components or parts interact dynamically in several different ways based on rules within that system, with possible interactions independent of any high-level instruction. True to this definition, as the number of connected devices in the home increases, available permutations and combinations for device chaining skyrocket, creating both new opportunities and unknown attack surfaces.

Common Protocols Used in Smart Homes

In this section, we introduce some of the protocols widely supported in automation servers.

MQTT

Message Queuing Telemetry Transport (MQTT) is a publish-and-subscribe messaging transport protocol over asynchronous communication.¹⁸ Simple and lightweight, MQTT is used for machine-to-machine (M2M) communication and IoT devices. MQTT brokers can encrypt using SSL (TLS), but encryption is often disabled in many cases. There are three roles in MQTT's general setup:

- **Broker:** Handles messages between MQTT clients called Publisher and Subscriber.
- **Publisher:** Sends messages to the Broker. The Publisher doesn't need to know about Subscribers.
- **Subscriber:** Receives messages from the Broker and uses the information called "Topic" to control which of the Publisher's messages are received.

Z-Wave

Z-Wave is an interoperable wireless communication protocol for devices that require low power and longtime operation such as home automation and sensor networks.¹⁹ Because it uses the 920MHz band in many regions, it has a relatively stronger shielding than high-frequency bands like Wi-Fi.

ZigBee

ZigBee is an IEEE 802.15.4-based short-range wireless communication protocol.²⁰ It can be used for home automation or industrial monitoring and for controlling applications. It is low power and has a short transferable distance and very low transfer speeds. ZigBee uses a 16-bit short addressing mode, which allows a single ZigBee network to support up to 65,536 nodes.

Bluetooth, Bluetooth Low Energy

Bluetooth is a wireless technology standard that uses the 2.4GHz band for exchanging data over short range.²¹ Bluetooth Low Energy (BLE) is available for Bluetooth version 4.0 and up, and major operating systems, including smart phone OSs, support BLE. Wearable devices and, recently, IoT hardware use BLE as their radio communication module to connect to a smart home gateway.

EnOcean

The EnOcean protocol is an international standard (ISO/IEC 14543-3-10) for wireless communications. It can generate electricity using energy harvesting technology and use that electric power to transmit data from its sensors.²² EnOcean switches transmit a unique 32-bit ID and have support for 128 AES encryption with rolling code, which prevents an attacker from brute forcing signals. The standard EnOcean module can transmit signals up to 300 m (free field) with only 50 microWS (μWs).

The table below shows the protocols supported by the major open-source IoT automation servers.

| | OpenHAB | PiDome ²³ | Domoticz | Home Assistant | FHEM | MajorDoMo ²⁴ | Mycontroller ²⁵ | Pimatic ²⁶ | Node-RED |
|-----------|---------|----------------------|----------|----------------|------|-------------------------|----------------------------|-----------------------|----------|
| MQTT | X | X | | X | X | X | X | X | X |
| Z-Wave | X | | X | X | X | X | | X | X |
| ZigBee | X | | | X | | | | X | |
| Bluetooth | X | X | X | X | | | | X (BLE) | |
| EnOcean | X | | X | X | X | | | | |

Table 1. Protocols supported by major open-source IoT automation server

IoT Automation Platforms

At the heart of the smart home is an emerging class of software products called IoT automation platforms, also known as home automation servers, which act as the brain for the smart home, functionally chaining devices to create smart applications. Thanks to a growing community of users who create and contribute plugins for new device support, these home automation servers have quickly outgrown their primary function and can now support a wide range of IoT devices, including home theaters, home monitoring tools, access controls, speakers, AI assistants, lights, climate controls, intercom/phones, door/window/motion sensors, environment monitors, garden care devices, and smart plugs.

There are three main types of IoT automation servers:

- **Local automation servers** — This is the most common type of automation server. They are generally installed as standalone servers that run locally on a Raspberry Pi, Arduino, Mac mini® computer, PC, etc. Communication with connected devices is done primarily via Ethernet using a patch panel. The automation server can also interact with secondary gateway servers, like the HA Bridge,²⁷ which emulates a Philips Hue light system, and with various wireless modules such as Wi-Fi, Bluetooth, Zigbee, Z-Wave, and EnOcean. All the automation rules are stored and processed locally on the server. For our research, we set up FHEM and Home Assistant servers to control all the test connected devices. We discuss these in greater detail later.
- **Cloud-based automation servers** — When services move to the cloud, there is an opportunity to add enhanced flexibility and functionality. The two platforms that we studied definitely do that. The first one, IFTTT, supports over 600 apps, devices, and companies from one interface.²⁸ IFTTT's goal is to enable users to push all their automation rules to the cloud and control their home from anywhere. It includes libraries of prebuilt applets (the platform's term for smart applications) to help users quickly and easily add new functionality. It also has the added advantage of being able to integrate devices with popular web services such as Spotify, Facebook, Instagram, and Uber, thus providing a rich user experience.

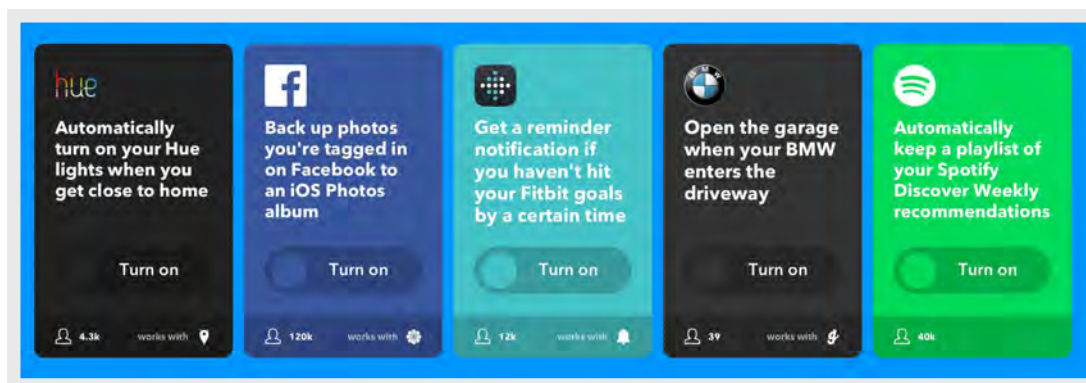


Figure 1. Screenshot of the IFTTT homepage showing controls for multiple smart applications²⁹

The only downside with IFTTT that we theorized was that too much personal data and control of the smart home was being outsourced to a third-party site. Relinquishing this level of control may be uncomfortable for some users especially with data breach incidents becoming commonplace. The second cloud platform that we studied can either be run in the cloud or run locally. Node-RED is a browser-based flow editor tool³⁰ for wiring together hardware devices, application programming interfaces (APIs), and online services. Node-RED can be run on standalone devices such as Raspberry Pi or can be fully deployed to cloud services like Amazon Web Services, Microsoft® Azure™ platform, Sense Tecnic FRED, and IBM Cloud™ platform. We found Node-RED to be very interesting because it can be used to build automation flows for both smart homes and industrial processes.

- **Virtual assistant automation servers** — This is pretty much the second-generation smart home Li et al. described in their paper,³¹ although the AI implemented in these devices is very rudimentary and with limited functionality. The three most popular virtual assistants (and corresponding devices) in the market today are Amazon Alexa (Amazon Echo™ speakers), Google Assistant Google Home™ smart speakers and Apple Siri (Apple® HomePod™ speakers).³² Third-party device manufacturers implement API integration with Amazon, Google, and Apple API kits, thus enabling their devices to be controlled by virtual assistants. These virtual assistants can also be used to create fairly complex automation workflows, but nowhere near as complex as workflows programmable using FHEM, Home Assistant, etc.

Smart Home Labs

For this research, we set up two complete smart homes: 1) a FHEM-controlled smart home in Germany and 2) a Home-Assistant-controlled smart home in the U.S. The German lab is a purpose-built smart home, whereas the U.S. lab is a bolt-on smart home. In this section, we provide high-level overviews of our smart home setups.

A. Smart Home Setup in Germany



Figure 2. Smart home setup in Germany

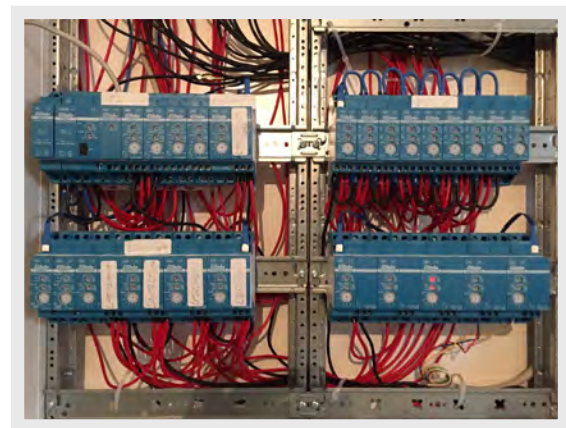
The smart home in Germany has more than 70 devices controlling everything, from light switches, doorbell, cameras, and phones to presence detection capabilities in devices. The key features are as follows:

- The house is connected to the internet using a FRITZ!Box™ cable modem, which also provides wireless internet for all devices in the house. FRITZ!Box has built-in support for Digital European Cordless Telecommunications (DECT), which is used by the cordless phones. The smart plugs, used to connect regular appliances like washers and dryers, also support DECT and thus talk directly to the FRITZ!Box.
- A network switch located in the basement is the main patch panel for the house. It connects the FRITZ!Box, Network Access Storage (NAS), EnOcean gateway, Raspberry Pi (RPI) running FHMM, RPi running HA-bridge, and the control panel for the front doorbell camera.

- HA-bridge is set up on an RPi connected to the network switch. HA-bridge emulates Philips Hue bulbs and essentially creates virtual switches (or fake Hue bulbs) that can be voice-controlled by an Amazon Echo device (Alexa).³³ Any device communicating using HA-bridge can be controlled using the virtual assistant Alexa.
- The Hue bulbs can be controlled using Alexa as well as using EnOcean switches. All door and window contact sensors, window shade controls, heating and air-conditioning controls also use EnOcean switches. EnOcean switches are energy-harvesting wireless switches that work up to a range of 30 meters inside buildings.³⁴ These switches transmit a unique 32-bit ID and support 128 AES encryption with rolling code, which prevents an attacker from brute-forcing signals.
- The RPi running FHEM has a Bluetooth® module which checks if a Bluetooth transmitter is present inside the house. It is used for presence detection.
- The front doorbell camera and other security cameras installed around the house can livestream video in a browser. The front doorbell camera also has motion detection and can be configured to send alert messages when motion is detected inside a specified visual frame of interest.
- FHEM communicates with all of these devices over Ethernet via the network switch. We can create custom rules tying different devices together to create smart applications. FHEM reads signals from all the connected devices as rule trigger conditions and sends out commands to the same devices via Ethernet. FHEM can also send SMS text alerts to preset mobile phone numbers.



Network switch located in the basement is the main patch panel and communication gateway for all Ethernet-connected devices.



EnOcean receiver array installed in the basement.

Figure 3. Network and EnOcean installed connections

We programmed a couple of smart applications in FHEM, two of which are shown in Figure 4.



Figure 4. FHEM rules for certain smart applications

Rule 1 turns on the patio lights after dark if the patio doors are opened. FHEM polls if the contact switches for either of the two patio doors are open or not. If open, it then checks if the lumen value reading is less than 500, signifying that the sun has set and that it is already evening. If the doors are opened and the lumen value is detected to be less than 500, FHEM turns on the patio lights for 10 minutes. The lumen value is read from the front doorbell camera.

Rule 2 checks if all the doors in the house are closed. If an EnOcean switch (installed beside the front door) is double pressed, FHEM checks if all six of the door contact switches are closed. If any of the contact switches is found open, indicating that a door is open, FHEM sends an SMS to the homeowner's mobile phone reporting which door has been left open. If no door is found open, then no SMS will be sent. The goal of this rule is to avoid the need to go around the house and manually check if each door is closed, that is, to make the entire process more efficient.

B. Smart Home Setup in the U.S.

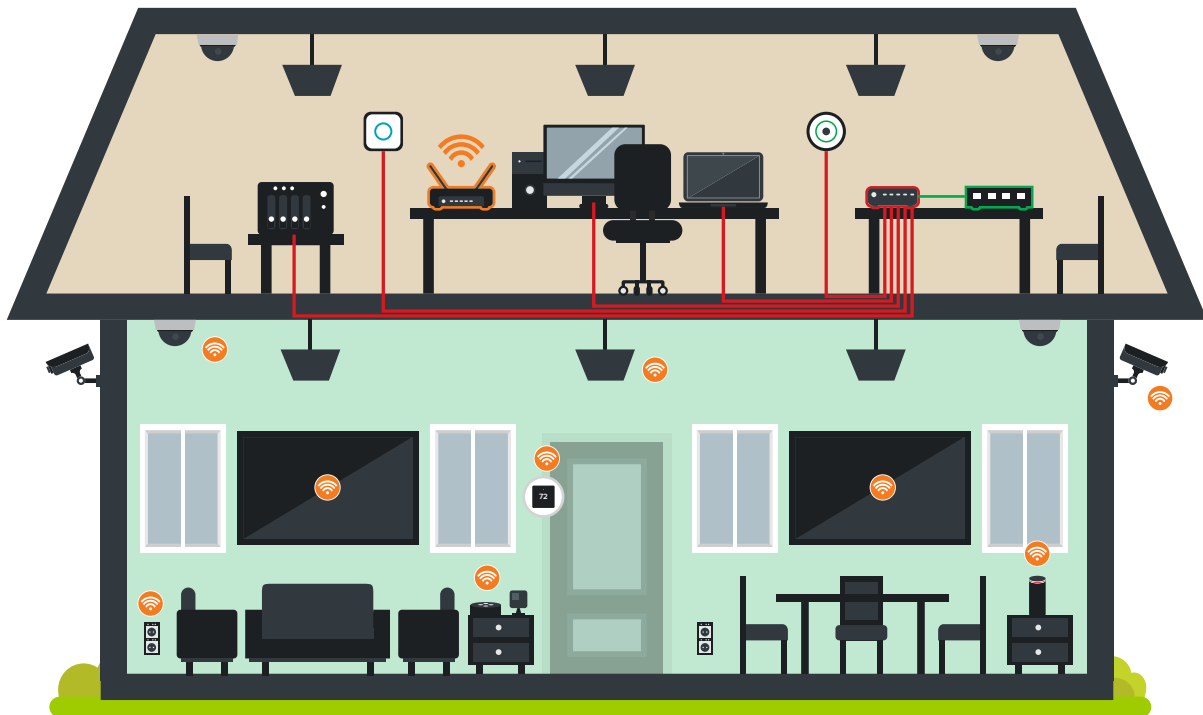


Figure 5. Smart home setup in the U.S.

The smart home setup in the U.S. has around 30 devices with similar functionality to those in the German smart home. Major additions that are not included in the German smart home are a thermostat with motion sensors, different types of cameras, and more smart plug adapters. The main goal of the U.S. smart home was to emulate the “bolt-on” style of smart homes that are typically found in the North American region. The U.S. smart home does not use special protocols like EnOcean but relies solely on Ethernet and Wi-Fi, and Zigbee for the Philips Hue Bridge to communicate with the light bulbs and switches. The key features of the smart home setup are as follows:

- An RPi running Home Assistant, an open-source home automation platform that is popular in the U.S. and around the world.
- An RPi running UniFi® Controller software that also aids in presence detection based on devices connected to the Wi-Fi. Multiple types of routers can be used for the setup, or Bluetooth like we did for the smart home in Germany.
- Ring® security cameras that act as motion sensors within Home Assistant.
- Amcrest security cameras that also act as motion sensors.
- An Ecobee thermostat that can be controlled via Home Assistant and can also act as a motion sensor for rooms that have remote sensors and the wall unit itself.

- Samsung® and Vizio smart TVs that use APIs to control the TV.
- A NAS to store footage from cameras and to offload logs to and from the RPi.
- Both a Google Home and an Amazon Echo are present in this lab for voice automation. Unlike in the case of the smart home in Germany, these function outside of Home Assistant and control items autonomously. Only the status changes related to the two devices are reflected in Home Assistant.
- Sonos speakers that are used as “home alarms” and for other types of output from Home Assistant.
- Roku® players are also connected to and controlled by Home Assistant.

The image displays two screenshots of the Home Assistant graphical user interface (GUI) for configuring an automation rule. The top screenshot shows the 'Triggers' section for a rule named '99.01 Test Wemo On'. It includes a description, a list of triggers (currently empty), and an 'ADD TRIGGER' button. The bottom screenshot shows the 'Action' section for the same rule. It includes a description, a list of actions (currently empty), and an 'ADD ACTION' button. The action configuration shows 'Call Service' with service 'switch.turn_on' and data {'entity_id': 'switch.wemo_mini'}.

Rule in Home Assistant graphical user interface (GUI) for configuration

```
- action:
  - data:
      entity_id: switch.wemo_mini
      service: switch.turn_on
      alias: 99.01 Test Wemo On
      condition: []
      id: '1520211490913'
    trigger:
      - at: 06:30:00
        platform: time
```

YAML file for configuration

Figure 6. GUI and YAML file for configuration

The rule here shows a basic configuration inside of the web UI to turn on a Belkin Wemo® switch at 6:30 a.m. local time. Home Assistant allows for basic configurations in the web interface making it very user-friendly. Advanced users can edit the .yaml configuration files. This allows them to create complex rules such as turning on the same switch from Monday to Friday, which requires the addition of condition fields.



```
- condition: time
  weekday:
    - mon
    - tue
    - wed
    - thu
    - fri
```

Figure 7. Configuration showing condition for days of the week

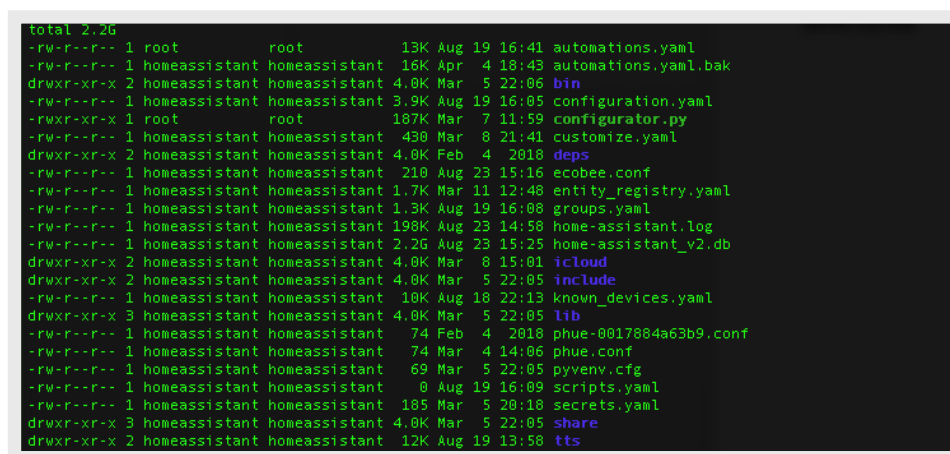
More complex rules can include things that involve detection from any of the motion sensors inside the smart home. In fact, one rule we set plays a doorbell sound on the Sonos speaker when it detects motion in the Ring Doorbell and the owners' phones are inside the house (this rule could also be used to play any sound). Another rule that we tested plays the sound of a dog barking if the owners' phones are not inside the house and Ring detects motion.

Both bolt-on and purpose-built homes carry their own advantages and disadvantages. However, we described these homes not to compare security risk levels but to demonstrate the level of complexity we were working with. This means that both homes are equally susceptible to the threat scenarios we have found using both setups. It is more the level of complexity, based on the number of devices and actions in a CIE, that would determine the risk rather than simply the type of smart home setup.

3. Attacks Against Complex IoT Environments

Complexity is the new enemy, and attacks against the logic layer are the new big headache. The main focus of smart home security research today is looking at individual components of a complex IoT environment and how singular devices might be attacked. To enable IoT devices to simplify and improve our lives, automation rules are becoming more and more complex — but this opens up new attack avenues that haven't been explored in depth before. Recently, we released a paper³⁵ that was a security analysis research on MQTT and Constrained Application Protocol (CoAP). And before that, Avast³⁶ had also published a blog about how MQTT-related flaws could be abused to attack smart homes. Both researches looked into M2M protocols, which is only a single element inside of a CIE, but did not discuss how some of the collected information could be leveraged to affect the whole ecosystem.

With home automation servers such as Home Assistant exposed online and not configured securely, there are ways that an attacker can collect information about how the systems are configured and what automation rules control the house. In this section, we explore some of these vectors for smart attacks using Home Assistant as our test bed.

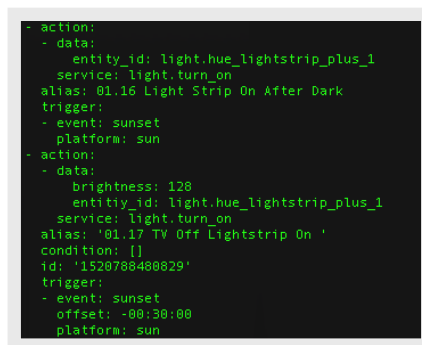


```
total 2.2G
-rw-r--r-- 1 root      root      13K Aug 19 16:41 automations.yaml
-rw-r--r-- 1 homeassistant homeassistant 16K Apr 4 18:43 automations.yaml.bak
drwxr-xr-x 2 homeassistant homeassistant 4.0K Mar 5 22:06 bin
-rw-r--r-- 1 homeassistant homeassistant 3.9K Aug 19 16:05 configuration.yaml
-rwxr-xr-x 1 root      root      187K Mar 7 11:59 configurator.py
-rw-r--r-- 1 homeassistant homeassistant 430 Mar 8 21:41 customize.yaml
drwxr-xr-x 2 homeassistant homeassistant 4.0K Feb 4 2018 deps
-rw-r--r-- 1 homeassistant homeassistant 210 Aug 23 15:16 ecobee.conf
-rw-r--r-- 1 homeassistant homeassistant 1.7K Mar 11 12:48 entity_registry.yaml
-rw-r--r-- 1 homeassistant homeassistant 1.3K Aug 19 16:08 groups.yaml
-rw-r--r-- 1 homeassistant homeassistant 198K Aug 23 14:58 home-assistant.log
-rw-r--r-- 1 homeassistant homeassistant 2.2G Aug 23 15:25 home-assistant_v2.db
drwxr-xr-x 2 homeassistant homeassistant 4.0K Mar 8 15:01 icloud
drwxr-xr-x 2 homeassistant homeassistant 4.0K Mar 5 22:05 include
-rw-r--r-- 1 homeassistant homeassistant 10K Aug 18 22:13 known_devices.yaml
drwxr-xr-x 3 homeassistant homeassistant 4.0K Mar 5 22:05 lib
-rw-r--r-- 1 homeassistant homeassistant 74 Feb 4 2018 phue-0017884a63b9.conf
-rw-r--r-- 1 homeassistant homeassistant 74 Mar 4 14:06 phue.conf
-rw-r--r-- 1 homeassistant homeassistant 69 Mar 5 22:05 pyvenv.cfg
-rw-r--r-- 1 homeassistant homeassistant 0 Aug 19 16:09 scripts.yaml
-rw-r--r-- 1 homeassistant homeassistant 185 Mar 5 20:18 secrets.yaml
drwxr-xr-x 3 homeassistant homeassistant 4.0K Mar 5 22:05 share
drwxr-xr-x 2 homeassistant homeassistant 12K Aug 19 13:58 tts
```

Figure 8. Screenshot of Home Assistant directory structure

As shown in the screenshot, inside of the U.S. Trend Micro Home Automation Lab, there are multiple .yaml configuration files. The two main configuration files are:

- **Configuration.yaml** — This file stores information on all devices communicating with the IoT automation server. This includes devices like Philips Hue hubs, Sonos speakers, WeMo plugs, Harmony® remotes, smart blinds, Tesla cars, thermostats, and even Roomba® robot vacuums.
- **Automations.yaml** — This file is where the actual automation rules are stored. These can be a combination of both simple and complex rules. An example of an automation rule that we set up in our lab is a Philips Hue light strip that turns on 30 minutes after sunset.



```
- action:
  - data:
      entity_id: light.hue_lightstrip_plus_1
      service: light.turn_on
      alias: '01.16 Light Strip On After Dark'
      trigger:
        - event: sunset
          platform: sun
    - action:
      - data:
          brightness: 128
          entity_id: light.hue_lightstrip_plus_1
          service: light.turn_on
          alias: '01.17 TV Off Lightstrip On '
          condition: []
          id: '1528788488829'
          trigger:
            - event: sunset
              offset: -00:30:00
              platform: sun
```

Figure 9. Configuration for turning on light strip after dark

There are multiple log files stored in Home Assistant, which keeps track of all device state changes inside the house. These can be viewed using the web UI or via log files that are stored on disk. If accessed, an attacker could use these files for reconnaissance, such as acquiring device usage schedules and even movement patterns inside the house. These files allow us to see if someone is moving around the house, turning on lights, turning off lights, and listening to speakers. If someone leaves the house, then we'll be able to see that person's device leave the network and when it rejoins. This is called presence detection, through which we can write rules about where people are with respect to their house.

A. Outsmarting Smart Locks

When we start pairing rules with presence detection, which can be done by allowing the home automation system to log in to the network router, the system pulls the information on what devices are currently connected to the home network and stores it as the file known_devices.yaml. Modifying this file will allow an attacker to add a trusted device to the smart home without the homeowner's knowledge or consent. We can theorize an attack scenario when this is coupled with a smart lock that requires someone to be present inside the house. In this scenario, an attacker adds a phantom device to the trusted devices list and sets the device status to be always "present" inside the house, thus fooling the lock's presence detection checks and keeps the door unlocked.

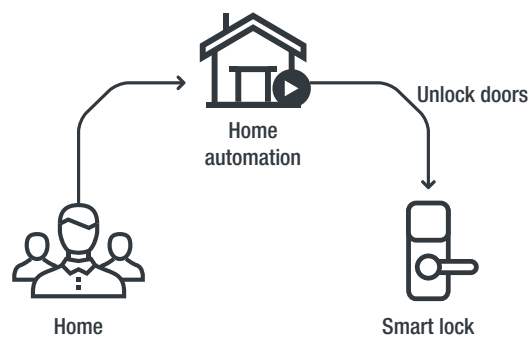


Figure 10. Logic flow of when the owners return home

Another example of a potential attack fooling smart locks would be to trick a smart lock to open for an attacker the way it would for one of the house owners. Say for simplicity's sake that the owner's device goes back online after being away, triggering an event to unlock the door so that the owner can walk into the house without using a key. The logic behind this will likely be the same logic someone will configure for their garage door, or any other door in the house as long as it has a smart lock. The owner can then manually lock the door after entering, and it will remain locked until one of the trusted devices leaves and returns home, at which point the process repeats. An attacker would only have to look at the automation configuration file with the known devices, insert a new trusted device the attacker owns into this file, and add it to the smart lock automation logic. When the attacker's device shows up at the front door, it is treated as if the owner has returned home and the smart lock unlocks the door. In this scenario, the attacker is not even modifying the smart home logic, merely adding an extra parameter (attacker's own device) that the owner is not aware exists and does not account for in the logic.

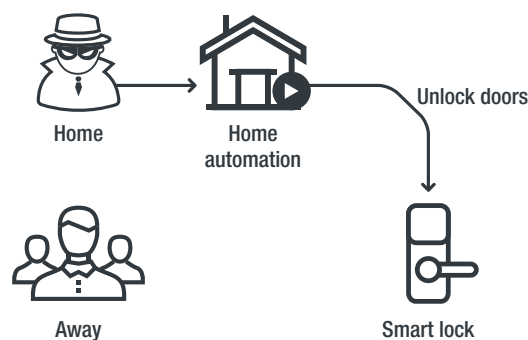


Figure 11. Attacker status is "at home" and the doors unlock

B. Spying Using Notifications

An altogether different attack scenario involves surveillance: an attacker could configure the home automation system to send messages to Slack or any other supported messaging platform. The attacker could then receive messages about activities inside the house. If there are cameras inside a house or even a place of business, an attacker could set up a monitoring system using the very devices inside the building. If motion is detected by any of the cameras, these cameras will capture and send snapshots directly to the attacker's Slack account.

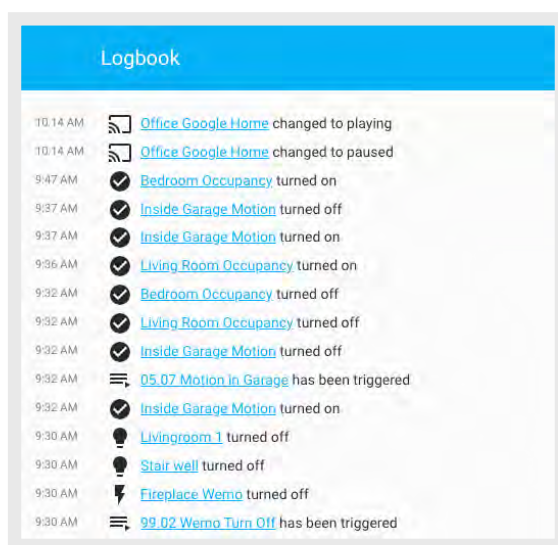


Figure 12. Graphical interface log of Home Assistant

As an example, say the attacker wants to write a rule for the garage motion sensor that will take a picture from the camera in the garage and send it to a Slack channel. The attacker could write a rule as shown in Figure 13.

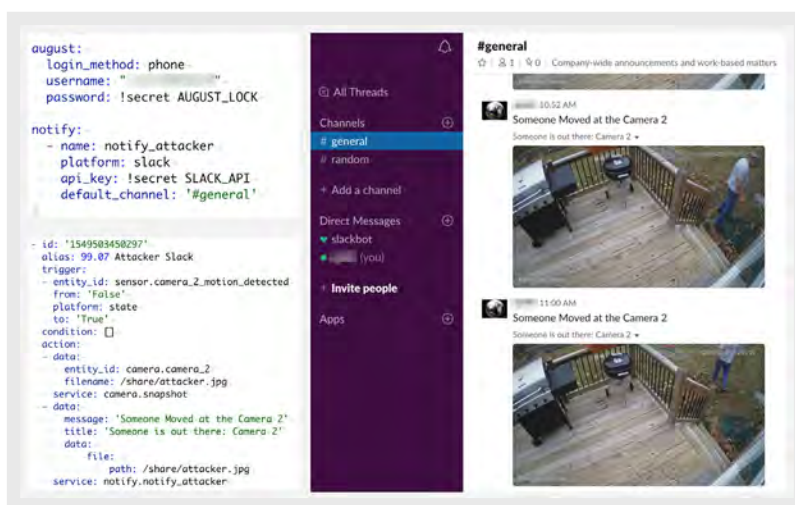


Figure 13. Example of a rule that an attacker could write to send a notification to Slack

One of the things that we tested in our lab environments was to see if changing the rules in multiple home automation servers would trigger a mechanism to alert users about the changes. There were no triggers, the rationale being, once a user has the automation rules in place, there is little to change and validate in the rules. Therefore, rules like the ones the attacker injects could go unnoticed indefinitely.

C. Controlling Speakers to Issue Commands

Let's say the owner of the complex IoT environment has a smart speaker connected in the home — the attack surface can now use sound. Most automation services have the text-to-speech (TTS) option enabled that allows users to play voice messages through any of the smart speakers right from the home automation server.

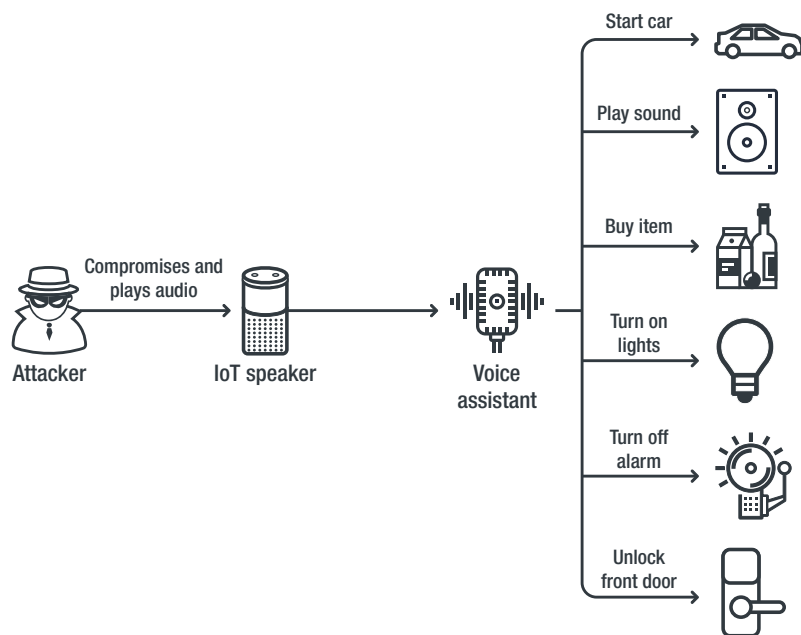


Figure 14. Attack chain for voice-controlled attacks

We covered these sound attacks in greater detail in our research *Sound of a Targeted Attack*.³⁷ In that research, we theorized an attack scenario wherein an attacker creates and hosts a sound file and uses the open APIs of the smart speakers to play the sound file from the specified URL. This time, in a CIE attack scenario, the attacker would not even need to host sound files, as the automation server can get the smart speaker to play the audio using TTS. If voice recognition is being used to validate commands, then the attacker can use existing software that can analyze someone's voice and clone it accurately. All the attacker needs to do is to upload and play the sound file and successfully bypass any voice recognition checks.³⁸

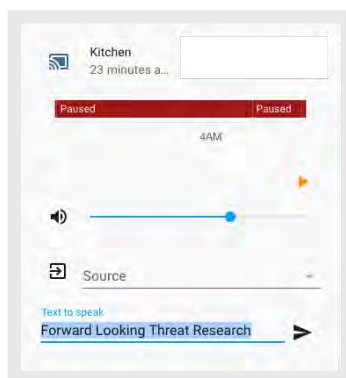


Figure 15. Example of text-to-speech order via Home Assistant

Of course, partnered with presence detection, the attacker could do all of this when no one is home, or even when someone is at home. By playing sounds at a frequency³⁹ that is inaudible to the human ear but perceptible to voice assistants like Alexa and Siri, the attacker can successfully issue commands.

Further expanding this threat landscape are cars, which are now being connected to home Wi-Fi networks and controlled by voice services such as Alexa and Google Assistant. In the future, when a car can start its engine, unlock its doors, or do other actions on voice commands, voice-based attacks will become a major issue. With the introduction of the “Hey Mercedes” voice trigger in the new Mercedes-Benz MBUX⁴⁰ car interface, voice threats to connected cars are one step closer to becoming reality.

D. Stealing Exposed Sensitive Data

In order for automation to function correctly, owners frequently need to hardcode personally identifiable information (PII), device username/password, and device API keys into configuration.yaml. If the automation server is exposed online, an attacker can download the full Home Assistant configuration and collect this sensitive data with ease. As part of our research, we scanned Shodan to see if there were any exposed automation servers. (The next section will detail the results of our scan.) We found an example of this scenario: the configuration file of a smart home in Norway (identifiable parts of the image have been deleted for privacy reasons), as shown in Figure 16.

```
1 homeassistant:
2   # Name of the location where Home Assistant is running
3   name: Home
4   # Location required to calculate the time the sun rises and sets
5   latitude: [REDACTED]
6   longitude: [REDACTED]
7   # Impacts weather/sunrise data (altitude above sea level in meters)
8   elevation: 23.10
9   # metric for Metric, imperial for Imperial
10  unit_system: metric
11  # Pick yours from here: http://en.wikipedia.org/wiki/List_of_tz_database_time_zones
12  time_zone: [REDACTED]
13  # Customization file
14  customize: !include customize.yaml
15
```

Figure 16. Exposed smart home configuration with PII

The user had input the exact location of the house (latitude and longitude) as well as the elevation and time zone. This data can be used to easily find the homeowner's address using a tool like Google Maps™ service.

```
204 switch:
205   platform: dlink
206   host:
207   username:
208   password:
209
210 # Text to speech
211 tts:
212   - platform: google
213
214 media_player:
215   - platform: plex
216     scan_interval: 1
217     use_episode_art: true
218     entity_namespace: 'plex'
219     use_custom_entity_ids: true
220
221 xiaomi_aqara:
222   discovery_retry: 5
223   gateways:
224     - key:
225
```

Figure 17. Exposed smart home configuration with critical keys

The user hardcoded the wireless router's username/password in the configuration.yaml. The user also hardcoded the key for xiaomi_aqara,⁴¹ which is a gateway plugin used to control Xiaomi Aqara-compatible devices such as wall switches, smoke detectors, door locks, motion sensors, gas leak detectors, etc. As shown in Figure 17, the user also has TTS enabled in the Home Assistant server, opening up the home to the sound attacks we described previously.

E. Interesting Logic Bugs

At the end of the day, the most severe attack against a smart home is the injection of logic bugs in its automation rules. IoT automation servers allow users to create complex automation rules that chain together disparate devices to create smart applications. As mentioned previously, once a user has set automation rules in place, there will be little to change and validate. Therefore, rules like the ones an attacker injects could go unnoticed indefinitely.

```

91 #MOTION SENSOR AUTOMATIONS
92 #-----
93
94 #Hallway motion sensor
95 - alias: Hallway motion light on
96 trigger:
97   platform: state
98   entity_id: binary_sensor.motion_sensor_
99   from: 'off'
100   to: 'on'
101 action:
102   - service: light.turn_on
103     entity_id: light.gateway_light_
104     data:
105       brightness: 5
106   - service: automation.turn_on
107     data:
108       entity_id: automation.MOTION_OFF
109
110 - alias: Hallway motion light off after a minute
111 trigger:
112   platform: state
113   entity_id: binary_sensor.motion_sensor_
114   from: 'on'
115   to: 'off'
116   for:
117     minutes: 1
118 action:
119   - service: light.turn_off
120     entity_id: light.gateway_light_
121   - service: automation.turn_off
122     data:
123       entity_id: automation.Motion_off

```

Figure 18. Configuration showing rules for triggering lights

The screenshots in Figure 18 and Figure 19 are from our Shodan scan. Figure 18 shows an automation rule for motion-sensor-triggered lights installed around the house. Logic bugs an attacker could introduce include setting all the sensors “off” no matter the trigger state, setting an incorrect entity_id (see Figure 18) so the sensors never work, changing the delays etc. If such changes were to be made, then security lights won’t turn on in cases like a break-in.

```

126 #DOOR SENSOR AUTOMATIONS
127 #-----
128
129 #window/door sensor
130 - alias: If the sensor is apart play alarm
131 trigger:
132   platform: state
133   entity_id: binary_sensor.door_window_sensor_
134   from: 'off'
135   to: 'on'
136 action:
137   - service: light.turn_on
138     entity_id: light.gateway_light_
139     data:
140       brightness: 5
141 - alias: If the sensor is together stop alarm
142 trigger:
143   platform: state
144   entity_id: binary_sensor.door_window_sensor_
145   from: 'on'
146   to: 'off'
147 action:
148   - service: light.turn_off
149     entity_id: light.gateway_light_
150

```

Figure 19. Configuration showing smart alarm rules

This is an interesting example of a smart application. The homeowner has created a smart alarm, albeit functionally incomplete, by combining two different devices: door sensors and lights. If an attacker modifies this logic, then the attacker can alter the triggered conditions and actions of the smart alarm, effectively disabling it without the homeowner’s knowledge.

F. Problems With Cloud-Connected Devices

After setting up an IoT automation server, home users may get a false sense of security that all their data and processes are now localized. This couldn't be further from the truth. Even if controlled by an automation server, many devices need to push commands to the cloud for processing and interpretation before execution. For example, the Alexa smart speaker listens for user queries triggered by announcing the keyword "Alexa." Once the keyword is announced, Alexa makes a record of the user's query, which it then sends to the Amazon cloud, where a service called the Alexa Voice Services interprets the user command and performs the requested action if Alexa has the "skill" installed.⁴² Alexa can control a wide variety of appliances, and yet Alexa is also integrated into the smart home using an automation server. If the Alexa service is interrupted, then the smart home will lose functionality.

This connectivity opens up an opportunity for an attacker to execute man-in-the-middle (MitM) or distributed denial-of-service (DDoS) attacks against certain classes of devices, essentially rendering them useless and severely affecting the smart home. MitM router attacks are becoming common⁴³; if an attacker compromises the router and alters the DNS, the responses devices get from the cloud can then be changed — effectively altering many of the parameters entering the logic flow of the house too. So even if the attacker cannot gain access to the Home Automation server with WRITE access, it's still possible to hack the logic through the said method.

In conclusion, as the migration towards full home automation moves steadily forward, brand-new attack vectors that take advantage of the IoT ecosystem are emerging yet everyday users are unaware of these threats. Soon, securing the home will take on a new meaning for both device manufacturers and security vendors.

4. Exposed IoT Automation Servers

Scanning the internet is important because security flaws can be quickly identified, discovered, and fixed before they are exploited. Traditional web search engines such as Google, Bing, and Yahoo! are great for searching for information and websites but are not so useful for searching for device metadata. For that task, we use Shodan. Unlike Google and Yahoo!, Shodan is a search engine specifically for internet-connected devices. It finds and lists devices and systems such as webcams, baby monitors, medical devices, home appliances, databases, etc.

The basic unit of data that Shodan gathers is the banner. The banner is textual information that describes a service on a device. In addition to the banner, Shodan also grabs metadata about the device, such as its geographic location, hostname, operating system, among others. Shodan uses a GeoIP database to map the scanned IP addresses to physical locations.⁴⁴ In short, Shodan collates and makes searchable both device metadata and banner information (services running) that internet-connected devices and systems are freely sharing with anyone querying them.

Using Shodan, we searched for IoT automation servers that were exposed on the public internet. We started by looking into platforms we used in our smart home labs: FHEM and Home Assistant. We also found other automation systems, both open source and commercial.

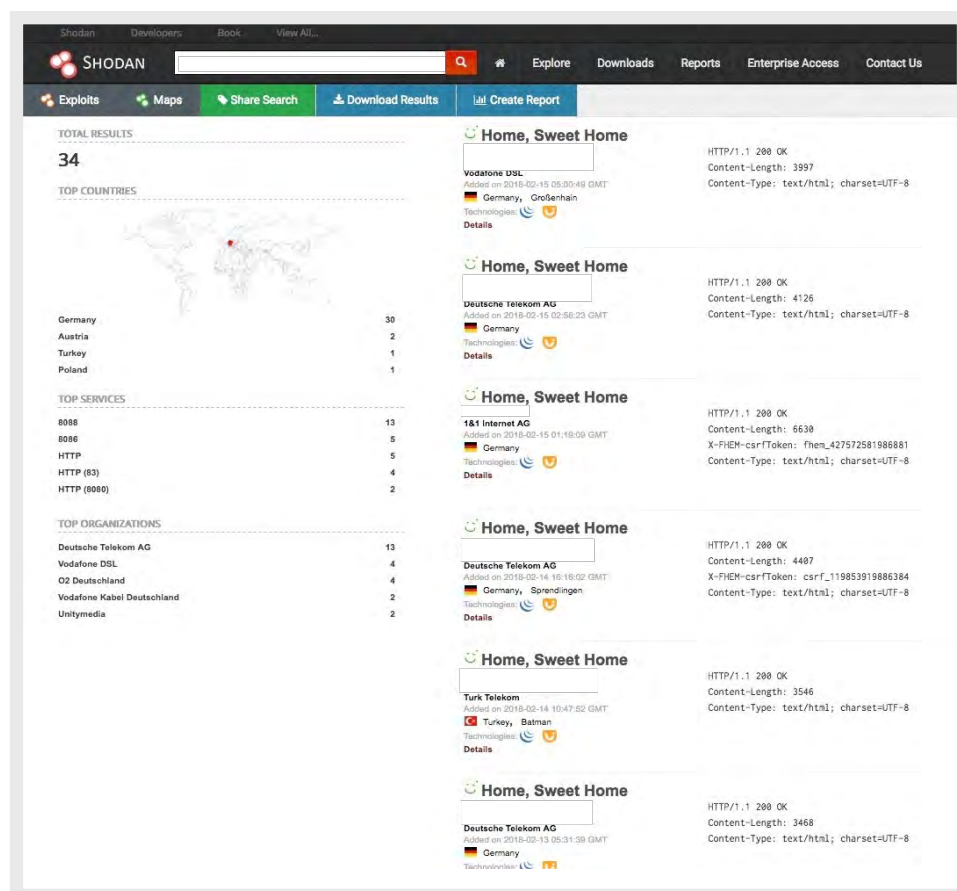
Exposed Open-Source Automation Servers

The key takeaways from exposed devices are that systems are freely sharing information with anyone querying them and systems may be accessible/interactable to anyone without requiring proper authentication. Open-source IoT automation servers will typically be set up by home users, many of whom lack the knowledge and skills to properly configure and protect their home networks. Adding to this conundrum is the fact that many of the open-source IoT automation servers don't have security features like password protection enabled by default, and don't prompt the users to enable security features either. An example of this is how FHEM requires only a series of non-trivial steps to enable password protection, and even then, it does not enforce setting up a secure password. Thus, we are left with many IoT automation servers sitting wide open on the public internet, waiting to get hacked.

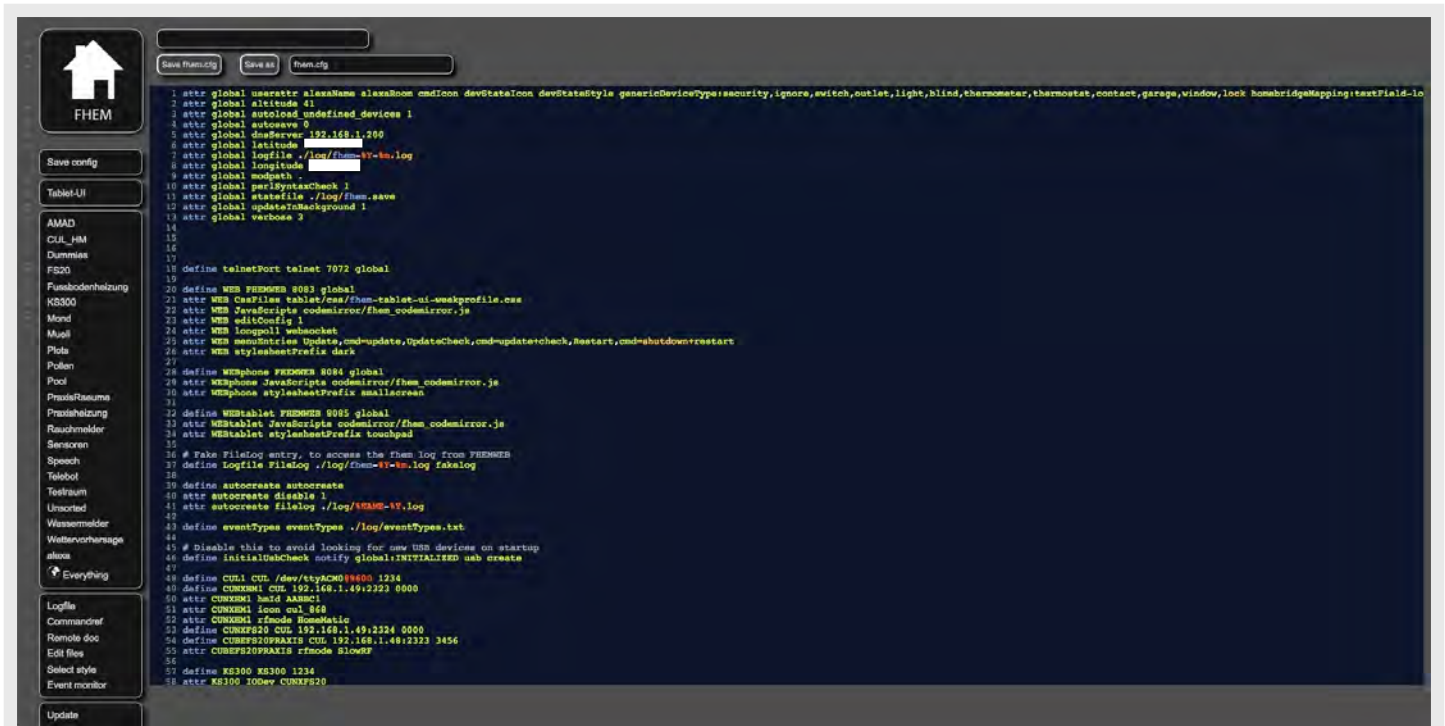
The two major security threat scenarios created by exposed automation servers are as follows: 1) an attacker can reprogram automation rules, steal hardcoded sensitive data, add new devices, infect devices with malware, harvest devices for botnets, etc. 2) an attacker can determine if people are present in the house or not, and can then disable the alarm and physically break into the house. In this section, we present three of the exposed automation servers we found using Shodan: FHEM, Home Assistant, and Node-RED. PII has been deleted from all images to safeguard privacy.

FHEM

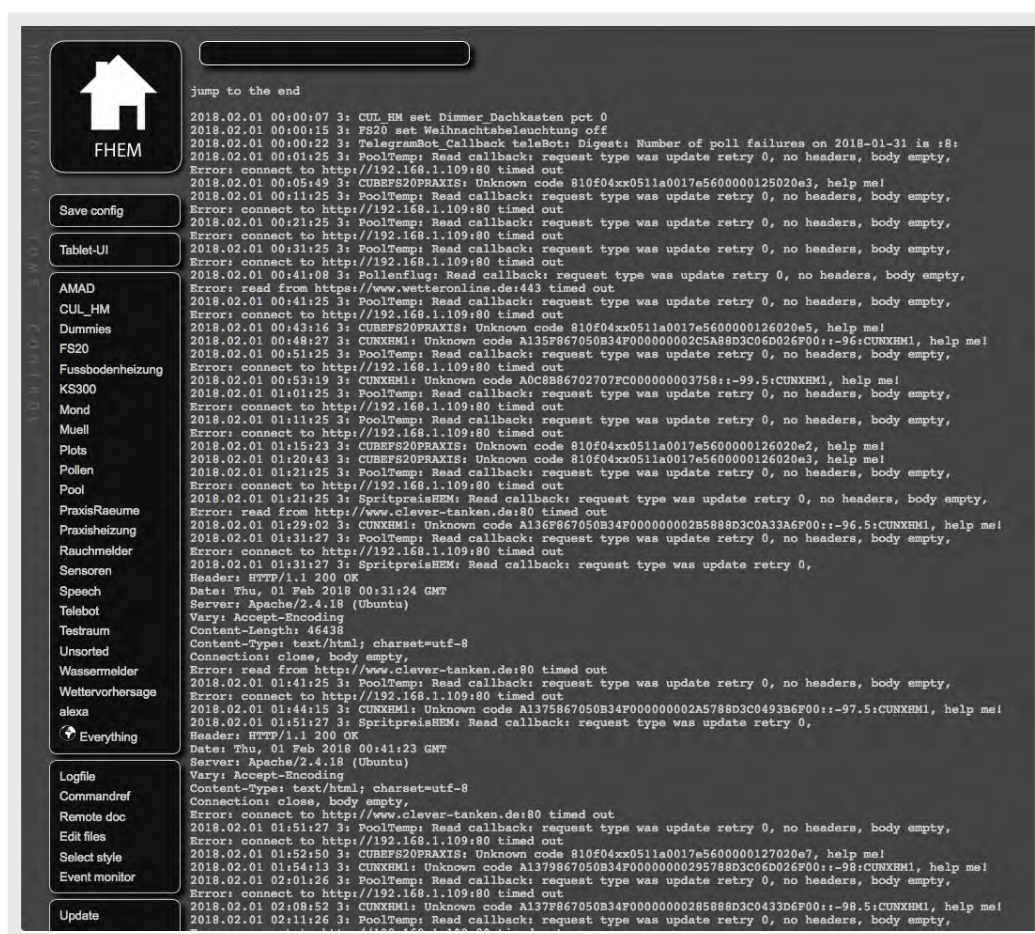
FHEM is a Perl server⁴⁵ for home automation that can be used to automate common tasks in the house, including controlling lights, shutters, heating, etc., and logging events or information like temperature, humidity, power consumption, etc. The program runs as a server on a dedicated device, for example, NAS, Raspberry Pi, PC, Mac mini, etc., and can be controlled directly via the web, smartphone, Telnet, or TCP/IP. FHEM has a Perl interpreter and its configuration file can be programmed in a Perl-like scripting language. FHEM is popular in Europe, especially in Germany, where there is a big community of FHEM users who develop FHEM plugins for new IoT devices. In our Smart Home Lab in Germany, we set up all home automation tasks using FHEM.



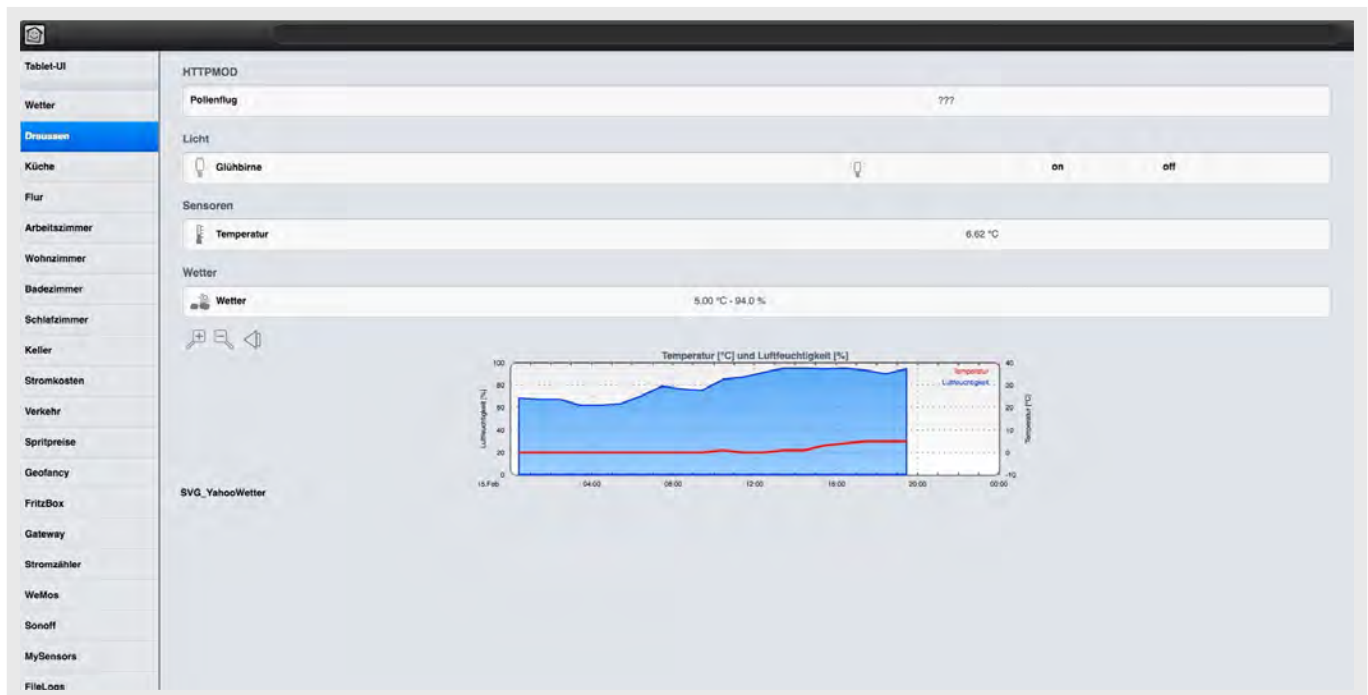
We found only a handful of FHEM servers exposed online, mostly in Germany and Austria.



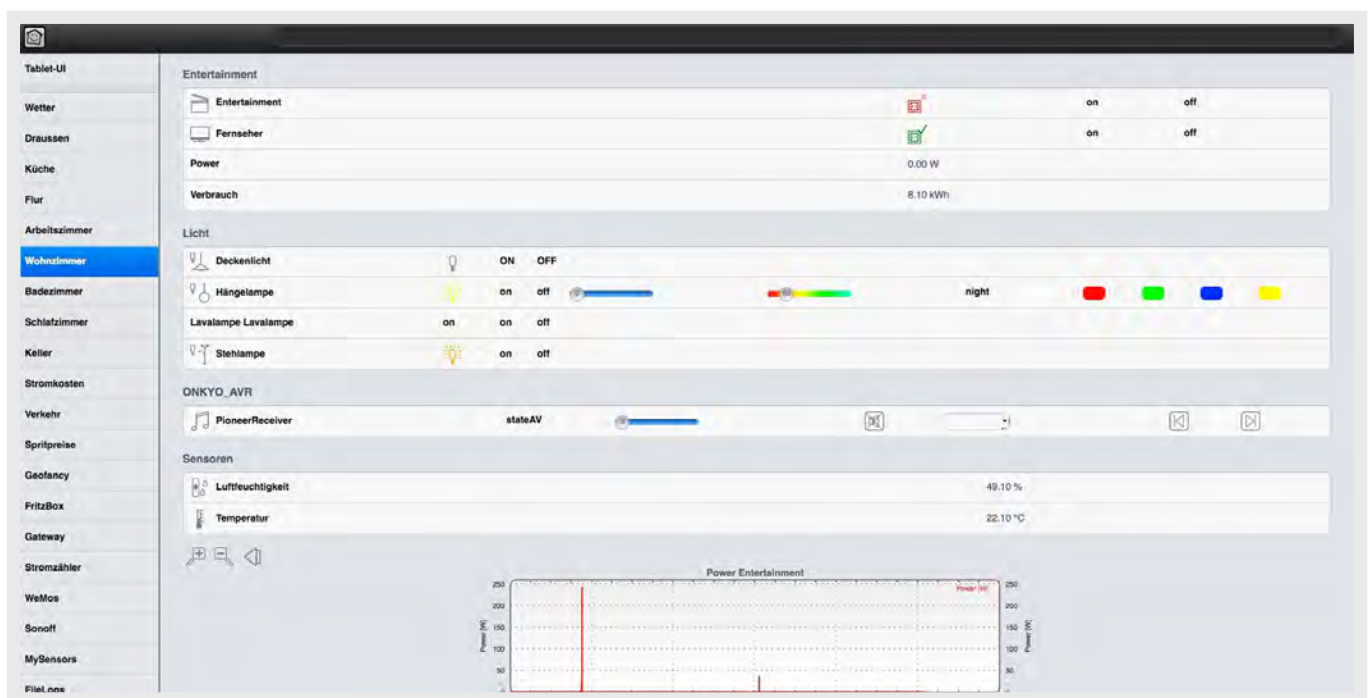
Some of these FHEM servers had their configuration files exposed. An attacker can rewrite the automation rules, collect hardcoded credentials, and get a sense of all the devices and where they are installed in the house. This makes it easy to plan attacks.



Detailed log files with records of all events triggered accessible from the same location as the configuration file.



FHEM server exposing readings and device status online.



FHEM server exposing details about lights, entertainment system, and sensor readings online.

| Gesamt-Stromkosten | |
|--------------------|----------------------|
| Gesamt | 133.60 kWh - 33.49 € |
| Jahr | 0.00 kWh - 0.00 € |
| Monat | 0.00 kWh - 0.00 € |
| Tag | 11.60 kWh - 2.91 € |

| KU-Stromkosten | |
|----------------|-------------------|
| Gesamt | 8.40 kWh - 2.11 € |
| Jahr | 0.00 kWh - 0.00 € |
| Monat | 0.00 kWh - 0.00 € |
| Tag | 3.50 kWh - 0.88 € |

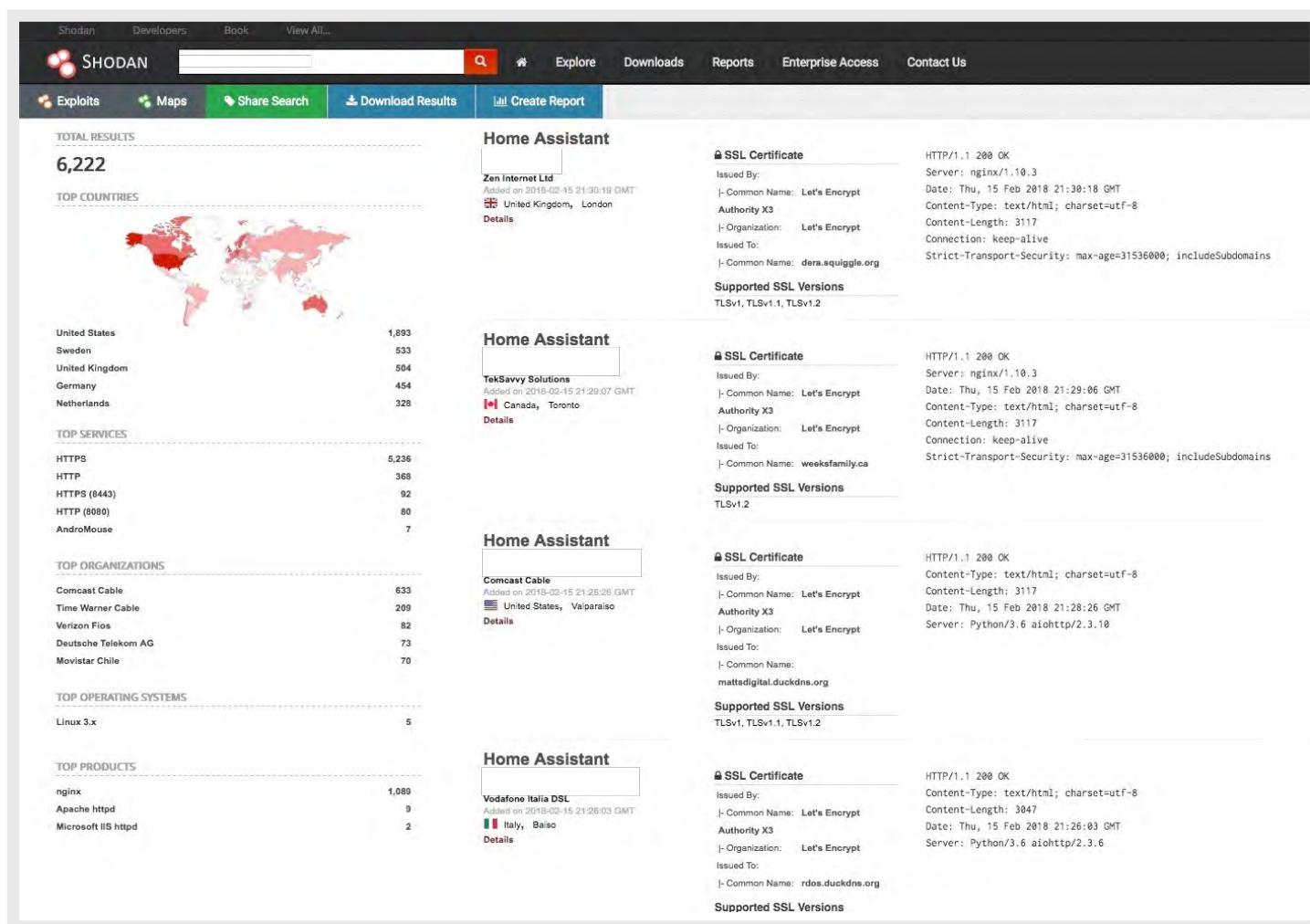
| WZ-Stromkosten | |
|----------------|----------------------|
| Gesamt | 125.20 kWh - 31.39 € |
| Jahr | 125.20 kWh - 31.39 € |
| Monat | 0.00 kWh - 0.00 € |
| Tag | 8.10 kWh - 2.03 € |

FHEM server exposing electricity usage values and the corresponding euro (€) amount billed.

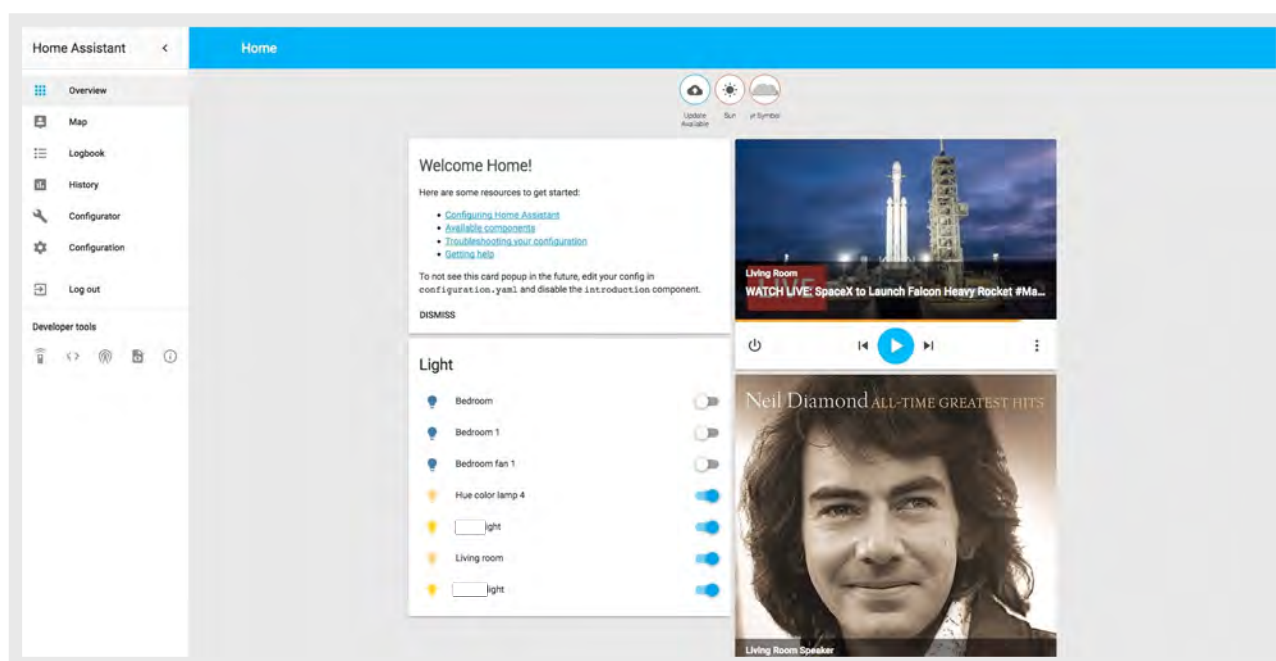
Figure 20. Screenshots of exposed FHEM servers

Home Assistant

Home Assistant is an open-source home automation server⁴⁶ that can be used to run all connected devices in the home from a single, mobile-friendly interface. Similar to FHEM, Home Assistant runs on a dedicated server like RPi or a local server, and all device data is stored locally and not in the cloud. Home Assistant out-of-the-box supports many of the popular IoT devices, including Nest® thermostats, Philips Hue products, Google Chromecast™ devices, Kodi® media boxes, Belkin WeMo switches, Ikea® smart bulbs, Plex media players, Vera™ products, etc. Thus, unlike with FHEM, in Home Assistant, it would not be necessary to install additional plugins. In addition to programming via the GUI, automation rules can be written using YAML. In our Smart Home Lab in the U.S., we set up all the home automation tasks using Home Assistant.



We discovered more than 6,200 Home Assistant servers exposed online. Majority of these servers are located in the U.S. and in Europe.



The good thing is that Home Assistant enforces password protection, and most of the exposed servers are password-protected. But there were still a fair amount of Home Assistant servers that were not password-protected.



The History feature shows the operational state/status of the devices over a specified time period.
An attacker can use this to observe and predict when people are at home or away.

```

59
60 # Weather prediction
61 sensor:
62   - platform: yr
63
64
65 switch:
66   platform: dlink
67   host: 192.168.0.6
68   username: admin
69   password: 205256
70
71 # Text to speech
72 tts:
73   - platform: google
74
75 media_player:
76   - platform: plex
77     scan_interval: 1
78     use_episode_art: true
79
80 xiaomi_aqara:
81   discovery_retry: 5
82   gateways:
83     - key: p
84
85 group: !include groups.yaml
86 automation: !include automations.yaml
87 script: !include scripts.yaml
88 scene:
89   - name: Livingroom normal
90     entities:
91       light.tradfri_bulb_e27_opal_1000lm_4:
92         state: on
93         transition: 2
94         brightness_pct: 100
95   - name: Livingroom dim
96     entities:
97       light.tradfri_bulb_e27_opal_1000lm_4:
98         state: on
99         transition: 2
100         brightness_pct: 50

```

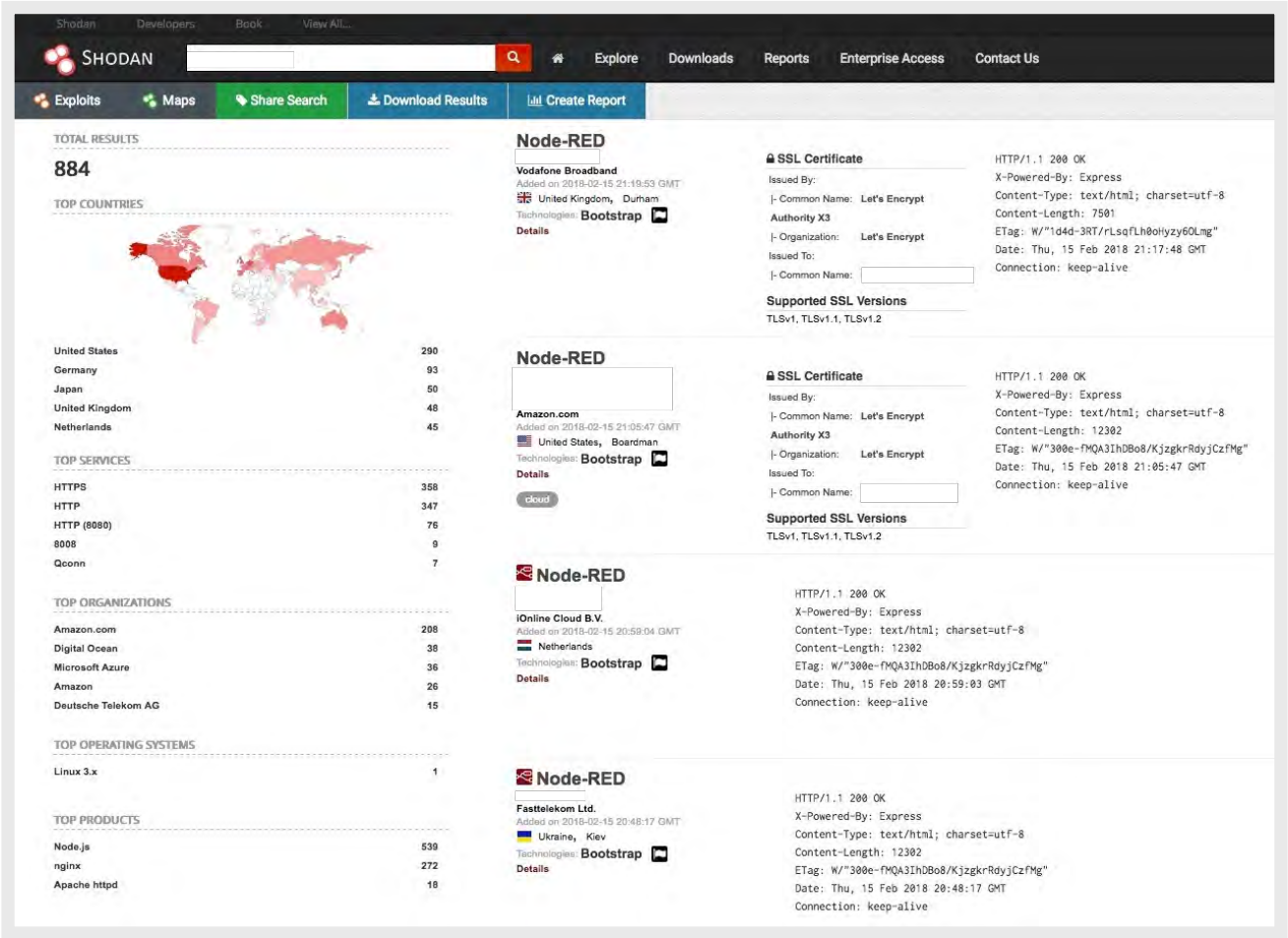
This yaml configuration file from Home Assistant contains the router username/password hardcoded as admin/205256. There are also device API keys hardcoded in the configuration. Exposed configuration files such as this leak a lot of sensitive data about the environment being controlled by Home Assistant.

Figure 21. Screenshots of exposed Home Assistant servers

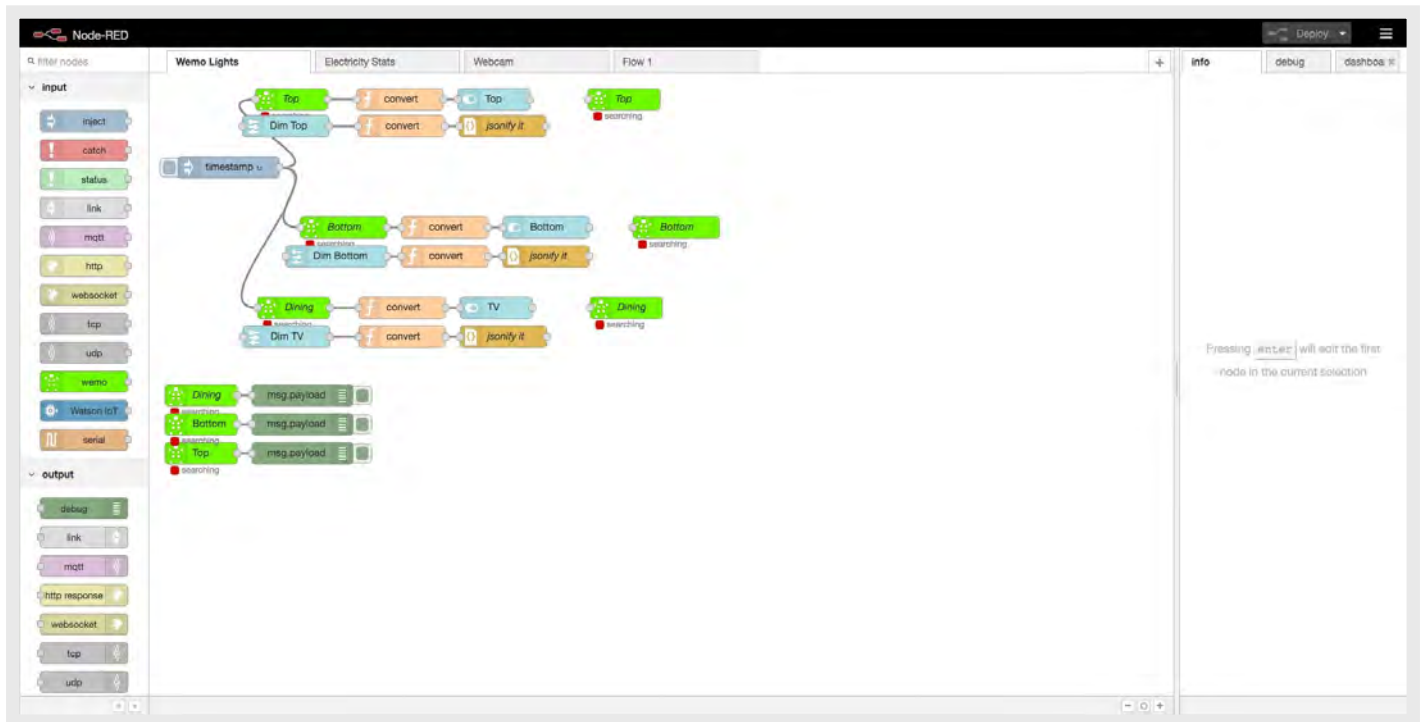
Node-RED

Node-RED is a flow-based programming tool⁴⁷ for wiring together hardware devices, APIs, and online services. It uses a browser-based flow editor and provides a host of nodes in its palette that can be deployed to the runtime flow with one click. A built-in library allows the user to save functions, templates, or flows to enable easy reuse. Node-RED is built using Node.js and created flows are stored using JSON. It can be run locally, on standalone devices like Raspberry Pi, or deployed to cloud services like Amazon

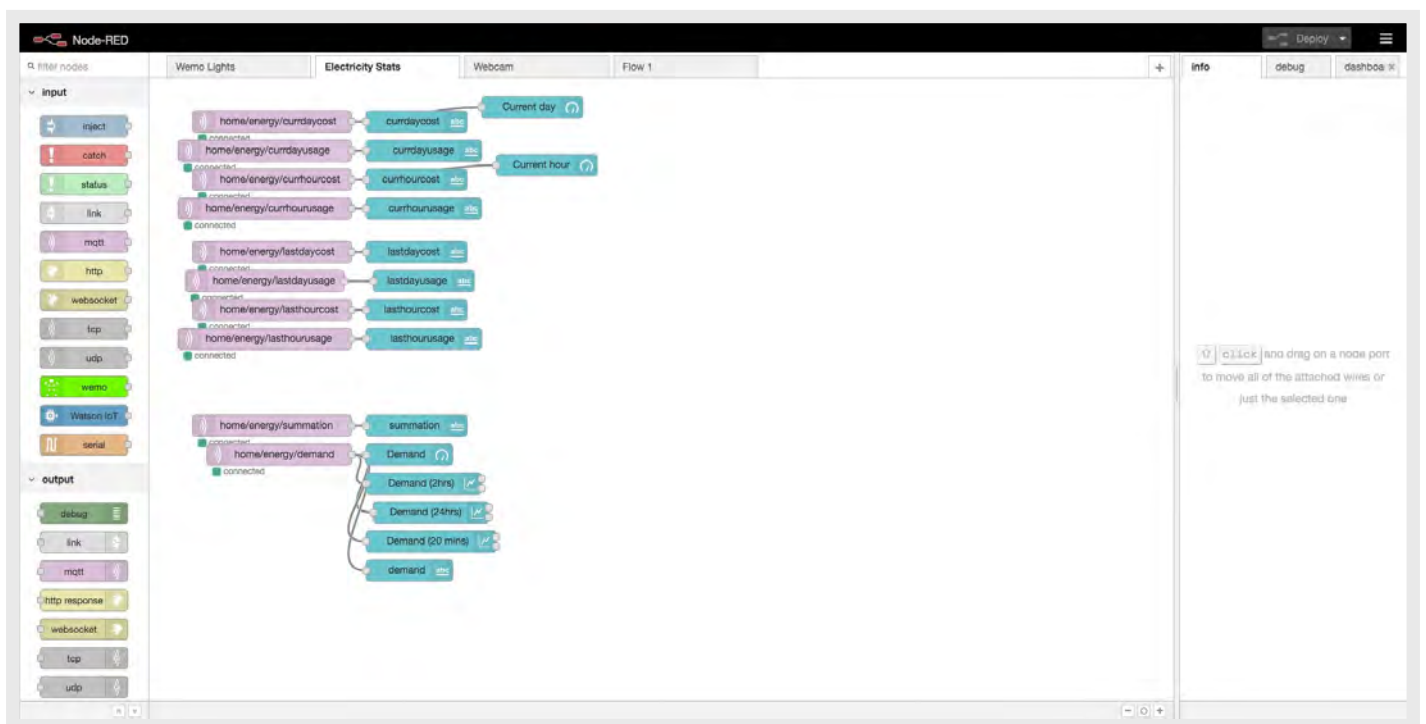
Web Services, Microsoft Azure, SenseTecnica FRED, and IBM Cloud. Node-RED is so interesting to us because it can be used to build automation flows for both smart homes and industrial processes. This crossover support between the IoT and IIoT spaces is the direction we think many of the current open-source IoT automation platforms will eventually follow.



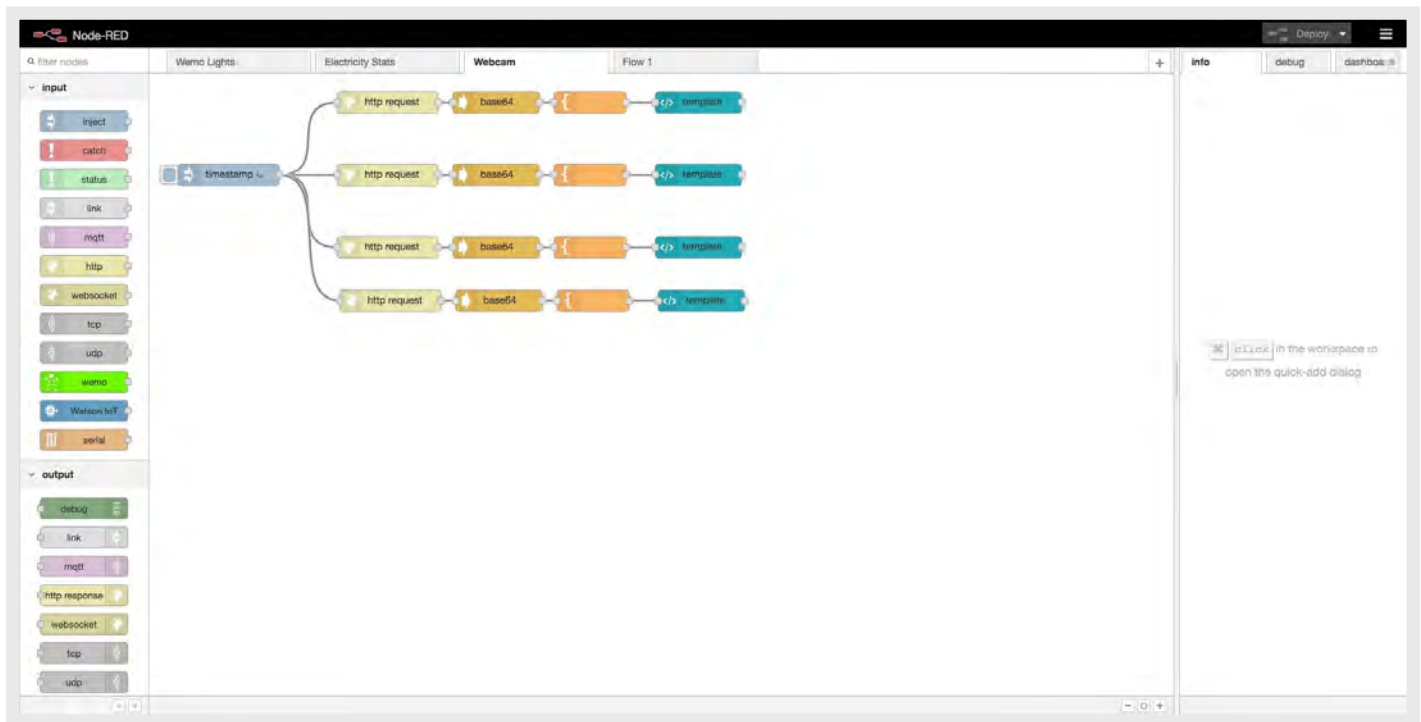
We found more than 880 instances of Node-RED exposed online. Most of the servers are located in the U.S., Germany, Japan, U.K., and the Netherlands.



Exposed automation flow for Wemo lights in a home in Australia.



Exposed automation flow for electricity stats in a home in Australia.



Exposed automation flow for a webcam in a home in Australia.

Figure 22. Screenshots of exposed Node-RED servers

Other Exposed Automation Servers

We presented only a small selection of the IoT automation servers that we discovered exposed in Shodan. FHEM and Home Assistant were of special interest to us because we used them to set up our smart home labs in Germany and the U.S. Node-RED caught our attention because it can be used for both home automation as well as industrial process automation, making it an interesting IoT-IIoT cross-platform product. It is our opinion that many of the open-source servers that we investigated will have their code bases forked and new servers created that also support industrial process automation. To do this, they need to support out-of-the-box protocols like MQTT, AMQP, STOMP, XMPP, WAMP, etc., which are used in industrial process communications. Other IoT automation servers that we discovered exposed in Shodan include Domoticz, openHAB, Loxone, Pimatic, Fibaro, etc.

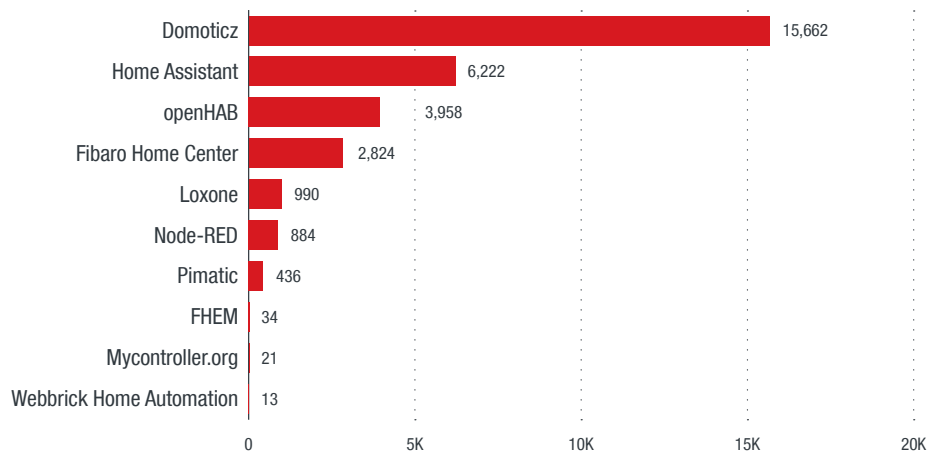
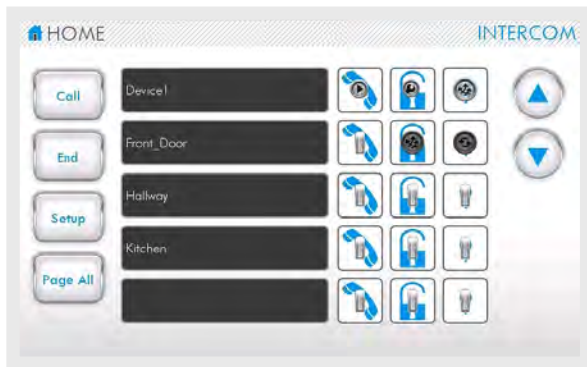


Figure 23. Exposed IoT automation servers found using Shodan

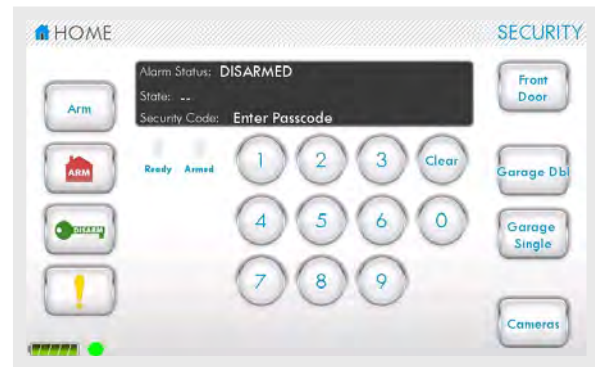
The most popular open-source IoT automation servers were Domoticz, Home Assistant, openHAB, and Fibaro Home Center. Countries with triple or quadruple digit numbers of exposed automation servers include Germany, the U.S., Japan, Sweden, the U.K., Netherlands, Australia, Canada, France, Norway, Croatia, Hungary, Italy, Austria, Belgium, Poland, Spain, Russia, Romania, and Czech Republic — basically, the industrial nations in Europe, North America, Australia, and Japan. Interestingly, we found automation servers also being used in Thailand, Vietnam, Chile, and Argentina. IoT automation is still in its early stages in many countries but looks to be catching on fast globally. To put these numbers in perspective, we need to mention that our collected data is purely based on exposed automation servers found in Shodan. The actual numbers will be far greater for three reasons: 1) Shodan has not crawled every exposed automation server out there; 2) not every automation server is exposed on the internet; 3) daily results fluctuate because of dynamic IP addresses.

Exposed Commercial Automation Servers

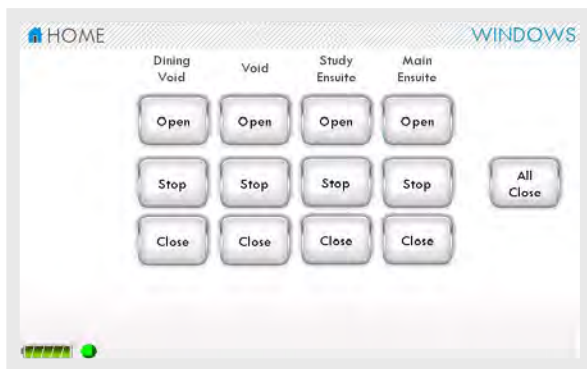
While searching for exposed open-source IoT automation servers in Shodan, we came across quite a few commercial home automation servers. While these commercial servers are not flexible like their open-source counterparts in integrating a wide variety of IoT devices into the environment, they still perform basic home automation/control functions but within a narrower scope. Users can only operate preinstalled devices, and they can create simple schedules for device operations — none of the complex logic layer programming that we find in open-source automation servers. Unfortunately, just like their open-source counterparts, these exposed systems are freely sharing information with anyone querying them, and systems may be accessible/interactable to anyone without requiring proper authentication. For functionality comparison, we have included exposed commercial automation servers in this section.



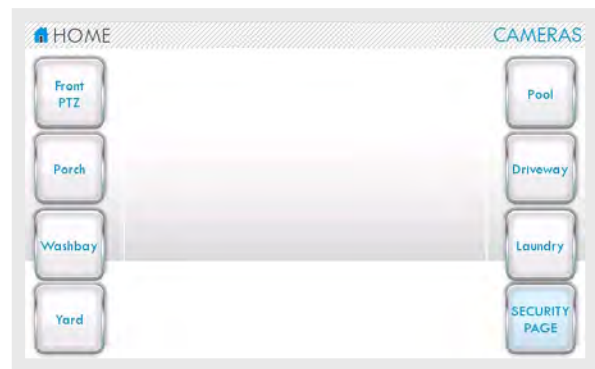
Intercom system controls for the entire house.



Home alarm system can be turned on/off from this menu. There are controls for the garage and front doors, as well as for installed cameras.



Window shade controls for the entire house.



Camera controls. The camera locations are labeled.



Log of the daily events. The user can also create custom schedules from this page.



Light controls for the entire house.

Figure 24. Screenshots of an exposed commercial automation server in a smart home

An attacker would not be able to carry out any meaningful smart attack against this home automation server by introducing logic layer bugs because there is NO user-programmable logic layer. On the other hand, this setup is still vulnerable because an attacker can spy around the house using the installed cameras to see if anyone is at home, disable the alarm system, open the motorized window shades, and allow someone to break into the house.



Home alarm system can be turned on/off from this menu.



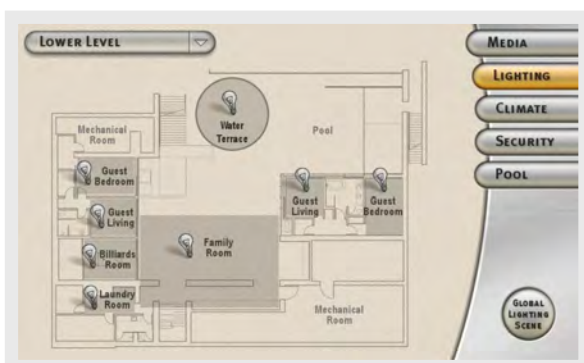
Spa and pool controls.



Climate control menu for the upper level.



Security control for the upper level.



Light controls for the lower level.



Alert menu showing a bathroom window is open.

Figure 25. Screenshots of an exposed commercial automation server in a different smart home

The commercial home automation system in Figure 25 is similar in functionality to the example in Figure 24. Again, an attacker would not be able to carry out any meaningful smart attack against this home automation server because there is no user-programmable logic layer. Commercial home automation systems like these two examples are steadily gaining popularity because they are professionally installed, maintenance is included in the service package, and they are simple to operate. But as IoT devices gain greater foothold in everyday households, and non-trivial device use cases emerge to enhance our daily lives, there will be a migration to user-programmable open-source IoT automation servers, especially as they become more user-friendly.

5. Protecting Complex IoT Environments

Today's society is adopting connected technologies at a faster rate than we are able to secure them. Every home is unique and hosts a wide variety of connected devices serving different functions. Unfortunately, there is no one-size-fits-all cybersecurity solution for these connected devices. Compared to the business environment, the connected home is unstructured, dynamic, and tends to be function oriented. The vast majority of people are either unaware or unconcerned about the potential security risks their exposed connected devices pose. The IoT ecosystem is multilayered and the risk factor of successful compromise increases with each additional layer.

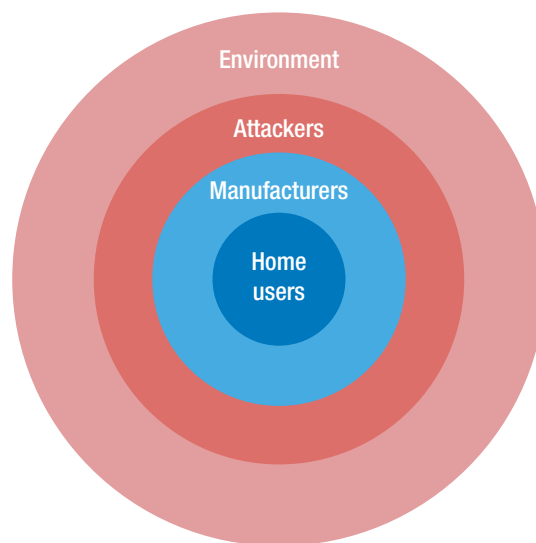


Figure 26. The IoT ecosystem risk factor increases when considering each layer of the IoT ecosystem⁴⁸

We have already described CIEs, smart applications, and smart attacks in depth in this paper. In this section, we provide recommendations to secure CIE. Many of these recommendations are common sense and cybersecurity experts repeatedly recommend them. When discussing how to secure connected devices at home, we also need to be mindful of three core IoT principles: 1) always online, 2) always available, and 3) easy to use. We also need to remember that the average household does not have a resident IT guru

who can secure everything connected to the network, and so enabling security features should be made as simple as possible. Our recommendations are as follows:

- Enable password protection on your devices. This is an easy option to enable on most connected devices that support passwords. It should be mandatory for smartphones, tablets, laptops, webcams, etc.
- Replace default passwords with strong passwords. Users routinely do not change the factory default passwords on their devices, and these default passwords can be easily discovered using any internet search engine. The other usual suspect is weak passwords that can be defeated using brute-force attacks or dictionary attacks.
- Change default settings. Many devices have all their supported services enabled by default, and many of those are unnecessary for day-to-day use for example, Telnet on webcams. If possible, the users should disable these unnecessary services. The only caveat is that advanced technical knowledge may be required to decide which services to disable and how to correctly disable them. We do not expect the average user to be knowledgeable about this; hence, it is up to the device manufacturers to make sure their devices are secure out-of-the-box.
- Do not jailbreak devices as jailbreaking can disable the built-in security features, making it easier for attackers to compromise the devices. Jailbreaking is popular especially with smartphones as this allows users with phones locked to a particular service provider to make them work for all service providers or in different countries.
- Do not install applications from unverified third-party marketplaces. Use only verified app marketplaces such as Apple's App Store, Google Play, and Amazon Appstore. Installing applications from unverified marketplaces is especially a big security risk on jailbroken iOS and Android devices. Apps installed from unverified third-party marketplaces can have backdoors built into them that criminals can use to steal personal information from the devices, or in the worst-case scenario take control of the device. The verified app marketplaces are not immune to hosting malicious ones, but the probability of that happening is small.
- Update the device firmware. This will fix known security vulnerabilities. On the flip side, there are many caveats with firmware updates: updating firmware for some devices might not be easy; the latest firmware is unstable and could introduce new bugs/issues; it is difficult to track each firmware update for numerous devices; updating the firmware of a device that is functioning correctly is not a priority; firmware update is not even possible.
- Enable encryption in both disk storage and communication platforms. Enable disk encryption for smartphones, tablets, laptops, and other devices to secure the data on disk even if the device is stolen. This is not a bulletproof solution but will secure the data on disk against theft even from the most skilled and resourceful hackers.

- Follow router specific best practices, some of which include enabling the router firewall, disabling WPS and enabling the WPA2 security protocol, and using a strong password for Wi-Fi access.
- Consider these other router security suggestions as well, though note that unfortunately these may limit device usage and functions: configure the router to limit device network access to set hours during the day/night; disable UPnP (this will limit the operations of connected devices); allow only a hardcoded list of device MAC addresses to access the network (the MAC address list will need constant updating).
- A more extreme measure is disconnecting the device from the network if internet access is optional for the device to function correctly. Though it goes against one of the core IoT principles of always being online, it reduces the possible attack surfaces for the entire system. For devices like a Wi-Fi bathroom scale, internet access is not required to measure body weight and is only required for the scale to send measured weight to an online portal that tracks daily changes in body weight and provides fitness suggestions.
- Make regular backups of the configuration and automation rule files of your IoT automation server. If possible, use a source code versioning or version control software to be able to revert code quickly as well as track historic changes. Also use a file integrity monitor to check that configuration and other files have not been tampered with.

Connected devices are an integral part of our daily lives. Ideally, device security should not affect the availability of the device and should be transparent to the user. As previously stated, there is no one-size-fits-all cybersecurity solution for connected devices. In addition to following the best practices and general guidelines we provided, users must be able to rely on the device manufacturers to enable strong security out of the box. Ultimately, we may need to rely on security by obscurity: connected devices hiding among billions of other connected devices online and avoiding being compromised by hackers.

6. Conclusion

IoT security will most likely become a multibillion dollar industry within the next few years as smart homes become more common and new IoT cyberthreats emerge. Upcoming IoT security products need to be very different from the traditional antivirus, antispam, web filtering, products, and so on that users are used to when they think of cybersecurity. The important thing to realize about future IoT security is its dynamic ecosystem with an unpredictable range of connected devices — each of which needs to be protected and users may need protection against.

- The first challenge is that IoT security products need to constantly discover what devices are connecting and/or disconnecting from the network. This device discovery needs to be done instantaneously and in a manner that doesn't overwhelm the network with unnecessary queries.
- The second challenge is that IoT security products must be able to positively identify what type a device is, for example, webcam, speaker, phone, light bulb, etc. This is easier said than done because devices are NOT mandated to respond to network queries in the same way. Popular devices like those manufactured by Google will probably announce themselves in a manner that is easily identifiable. However, cheap gadgets bought from ecommerce sites may not have announcement mechanisms, thus making it very difficult to identify them accurately. Adding to this conundrum are devices communicating via a bridge server such as HA-bridge. HA-bridge essentially creates a virtual switch that emulates a Hue light bulb, but the device connected to this virtual switch could be a smart speaker. Such a setup complicates the whole device identification process.
- The third challenge is that, once a device has been identified, the IoT security product needs to be able to analyze risks against the device and the risks posed by the device. In a CIE, the possible number of permutations and combinations of devices to create smart applications is massive. As more devices are added, that number exponentially increases. We have shown in this research that a CIE introduces unpredictable attack surfaces, bringing up questions like, How would it be possible to validate that a sound byte playing over Sonos is not instructing Alexa to disable the motion sensors around the house? Thus, future IoT security products need to be able to analyze as well as predict incoming threats against the CIE. This is all in addition to protecting against all the everyday threats like DDoS, MitM, zero-day, IoT malware, malware, unpatched vulnerability exploitation, and the like.

- The final challenge is protection, which is the primary purpose of an IoT security product. To reiterate, consumers expect three fundamental things from their connected devices: 1) always online, 2) always available, 3) easy to use. Break any one of these fundamentals and the whole user IoT experience will be destroyed. IoT security products need to be able to apply protection to the entire CIE in a way that will still preserve these three fundamentals and ensuring that the user does not feel IoT security as a burden.
- An altogether different challenge is that different classes of devices will require different levels and types of protection. As new device types are introduced, new protection rules will have to be deployed. Protecting one class of device should not compromise the functionality and security of another class of device, all the while NOT breaking the three fundamentals.

We conclude that IoT security is far from an easy problem to solve, but it is NOT an impossible one. The true challenge is trying to keep pace with the development of new IoT devices and the ever-evolving CIE. IoT automation compounds this problem by creating both new opportunities for functionality and unpredictable attack surfaces. The future of CIEs is going to be bright, exciting, and challenging all at the same time.

References

1. Intel. (n.d.) *Intel*. “Home Connectivity Reimagined.” Last accessed on 21 January 2019 at <https://www.intel.com/content/www/us/en/smart-home/connected-home/connected-home-solutions.html>.
2. Node-RED. (n.d.). *Node-RED*. Last accessed on 29 October 2018 at <https://nodered.org>.
3. EnOcean GmbH. (n.d.). *EnOcean*. Last accessed on 29 October 2018 at <https://www.enocean.com/en>.
4. Zigbee Alliance. (n.d.). *Zigbee Alliance*. Last accessed on 29 October 2018 at <http://www.zigbee.org>.
5. Silicon Laboratories. (n.d.). *Safer, Smarter Homes Start with Z-Wave*. Last accessed on 29 October 2018 at <http://www.z-wave.com>.
6. Bluetooth SIG Inc. (n.d.). *Bluetooth*. Last accessed on 29 October 2018 at <https://www.bluetooth.com>.
7. Rainieri Romera (11 February 2013). *Trend Micro Security Intelligence Blog*. “Home Automation and Cybercrime.” Last accessed on 5 February 2019 at <https://blog.trendmicro.com/trendlabs-security-intelligence/the-dark-side-of-home-automation/>.
8. Amazon.com Inc. (n.d.). *Amazon Alexa*. Last accessed on 29 October 2018 at <https://developer.amazon.com/alexa>.
9. Google. (n.d.). *Google Assistant*. Last accessed on 29 October 2018 at https://assistant.google.com/intl/en_ca.
10. Signify Holding. (n.d.). *Philips Hue*. Last accessed on 29 October 2018 at <https://www2.meethue.com/en-ca>.
11. FHEM. (n.d.). *FHEM*. Last accessed on 29 October 2018 at <https://fhem.de>.
12. Home Assistant. (n.d.). *Home Assistant*. “Automation.” Last accessed on 29 October 2018 at <https://www.home-assistant.io/components/automation>.
13. openHAB Community and openHAB Foundation e.V. (n.d.). *openHAB*. Last accessed on 29 October 2018 at <https://www.openhab.org>.
14. Domoticz. (n.d.). *Domoticz*. Last accessed on 29 October 2018 at <http://www.domoticz.com>.
15. Phil Muncaster. (6 December 2018). *Infosecurity* “Nokia: IoT Botnets Comprise 78% of Malware on Networks.” Last accessed on 21 January 2019 at <https://www.infosecurity-magazine.com/news/iot-botnets-78-of-malware-on/>.
16. Rita Yi Man Li, Herru Li, Cho Mak, and Tony Tang. (4 September 2016). “Sustainable Smart Home and Home Automation: Big Data Analytics Approach.” *International Journal of Smart Home*, 10(8), 177-187. Last accessed on 29 October 2018 at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2834497.
17. Steven Johnson. (2001). *Emergence: The connected lives of ants, brains, cities, and software*. New York: Scribner.
18. MQTT Org. (n.d.). *MQTT*. Last accessed on 30 October 2018 at <http://mqtt.org>.
19. Silicon Laboratories. (n.d.). *Safer, Smarter Homes Start with Z-Wave*. Last accessed on 29 October 2018 at <http://www.z-wave.com>.
20. Zigbee Alliance. (n.d.). *Zigbee Alliance*. Last accessed on 29 October 2018 at <http://www.zigbee.org>.
21. Bluetooth SIG Inc. (n.d.). *Bluetooth*. Last accessed on 29 October 2018 at <https://www.bluetooth.com>.

22. EnOcean GmbH. (n.d.). *EnOcean*. "Radio Technology." Last accessed on 29 October 2018 at <https://www.enocean.com/en/technology/radio-technology>.
23. PiDome. (n.d.). *PiDome*. "PiDome Home automation." Last accessed on 29 January 2019 at <https://pidome.org/>.
24. MajorDoMo. (n.d.). *MajorDoMo*. "MajorDoMo - free (open source) software for Smart home DIY." Last accessed on 29 January 2019 at <https://majordomohome.com/>.
25. MyController. (n.d.). *MyController*. "The Open Source Controller." Last accessed on 29 January 2019 at <https://www.mycontroller.org/#/home>.
26. Pimatic. (n.d.). *Pimatic*. "Pimatic Smart Home Automation Smart Home Automation for the Raspberry Pi." Last accessed on 29 January 2019 at <https://pimatic.org/>.
27. BWS Systems (7 August 2018). "Bwssystems/ha-bridge." Last accessed on 29 October 2018 at <https://github.com/bwssystems/ha-bridge>.
28. IFTTT. (n.d.). *IFTTT Platform*. Last accessed on 29 October 2018 at <https://platform.ifttt.com>.
29. Ibid.
30. Node-RED. (n.d.). *Node-RED*. Last accessed on 29 October 2018 at <https://nodered.org>.
31. Rita Yi Man Li, Herru Li, Cho Mak, and Tony Tang. (4 September 2016). "Sustainable Smart Home and Home Automation: Big Data Analytics Approach." *International Journal of Smart Home*, 10(8), 177-187. Last accessed on 29 October 2018 at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2834497.
32. Forbes Finds. (9 February 2018). *Forbes*. "Apple HomePod Vs. Amazon Echo Vs. Google Home: Which Smart Speaker Is Right For You?" Last accessed on 15 November 2018 at <https://www.forbes.com/sites/forbes-finds/2018/02/09/apple-homepod-vs-amazon-echo-vs-google-home-which-smart-speaker-is-right-for-you/#6e010ff47dec>.
33. BWS Systems (7 August 2018). "Bwssystems/ha-bridge." Last accessed on 29 October 2018 at <https://github.com/bwssystems/ha-bridge>.
34. EnOcean GmbH. (n.d.). *EnOcean*. "Radio Technology." Last accessed on 29 October 2018 at <https://www.enocean.com/en/technology/radio-technology>.
35. Federico Maggi, Rainer Vosseler and Davide Quarta. (4 December 2018). *Trend Micro*. "The Fragility of Industrial IoT's Data Backbone Security and Privacy Issues in MQTT and CoAP Protocols." Last accessed on 28 January 2019 at <https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/mqtt-and-coap-security-and-privacy-issues-in-iiot-communication-protocols>.
36. Martin Hron. (16 August 2018). *Avast Blog*. "Are smart homes vulnerable to hacking?" Last accessed on 30 October 2018 at <https://blog.avast.com/mqtt-vulnerabilities-hacking-smart-homes>.
37. Stephen Hilt (2017). *TrendLabs Security Intelligence Blog*. "The Sound of a Targeted Attack." Last accessed on 30 October 2018 at <https://documents.trendmicro.com/assets/pdf/The-Sound-of-a-Targeted-Attack.pdf>.
38. Samantha Cole. (8 March 2012). *Motherboard*. "Deep Voice Software Can Clone Anyone's Voice With Just 3.7 Seconds of Audio." Last accessed on 30 October 2018 at https://motherboard.vice.com/en_us/article/3k7mgn/baidu-deep-voice-software-can-clone-anyones-voice-with-just-37-seconds-of-audio.

39. Craig Smith. (10 May 2018). *The New York Times*. "Alexa and Siri Can Hear This Hidden Command. You Can't." Last accessed on 30 October 2018 at <https://www.nytimes.com/2018/05/10/technology/alexa-siri-hidden-command-audio-attacks.html>.
40. Daimler AG. (n.d.). *Daimler*. "Revolution in the Cockpit: Mercedes-Benz User Experience." Last accessed on 30 October 2018 at <https://www.daimler.com/innovation/case/connectivity/mbux-2.html>.
41. Home Assistant. (n.d.). *Home Assistant*. "Xiaomi Gateway (Aqara)." Last accessed on 30 October 2018 at https://www.home-assistant.io/components/xiaomi_aqara.
42. Richard Baguley and Colin McDonald. (4 August 2016). *CNET*. "Appliance Science: Alexa, how does Alexa work? The science of the Amazon Echo." Last accessed on 30 October 2018 at <https://www.cnet.com/news/appliance-science-alexa-how-does-alexa-work-the-science-of-amazons-echo>.
43. James Sanders. (17 April 2018). *TechRepublic*. "Why router-based attacks could be the next big trend in cybersecurity." Last accessed on 15 November 2018 at <https://www.techrepublic.com/article/why-router-based-attacks-could-be-the-next-big-trend-in-cybersecurity/>.
44. John Matherly. (2017). *Leanpub*. "Complete Guide to Shodan Collect. Analyze. Visualize. Make Internet Intelligence Work for You." Last accessed on 30 October 2018 at <https://leanpub.com/shodan>.
45. FHEM. (n.d.). *FHEM*. Last accessed on 29 October 2018 at <https://fhem.de>.
46. Home Assistant. (n.d.). *Home Assistant*. "Automation." Last accessed on 29 October 2018 at <https://www.home-assistant.io/components/automation>.
47. Node-RED. (n.d.). *Node-RED*. Last accessed on 29 October 2018 at <https://nodered.org>.
48. Martin Rösler. (3 November 2016). *Trend Micro*. "Securing Smart Homes." Last accessed on 30 October 2018 at <http://www.trendmicro.com/vinfo/us/security/news/internet-of-things/securing-smart-homes>.



TREND MICRO™ RESEARCH

Trend Micro, a global leader in cybersecurity, helps to make the world safe for exchanging digital information.

Trend Micro Research is powered by experts who are passionate about discovering new threats, sharing key insights, and supporting efforts to stop cybercriminals. Our global team helps identify millions of threats daily, leads the industry in vulnerability disclosures, and publishes innovative research on new threats techniques. We continually work to anticipate new threats and deliver thought-provoking research.

www.trendmicro.com



| research 